Computer persönlich

Christian Langfelder

BASIC OHNE PROBLEME

BAND 2: ÜBUNGEN

20 ausgewählte Routinen und Programme zum Veranschaulichen und üben allgemeiner Programmiertechniken auf cbm-Rechnern (cbm 8032)

BASIC ohne Probleme
Band 2: Übungen

Christian Langfelder

BASIC ohne Probleme Band 2: Übungen

20 ausgewählte Routinen und Programme zum Veranschaulichen und Üben allgemeiner Programmiertechniken auf cbm-Rechnern (cbm 8032)

Verlag Markt & Technik

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Langfelder, Christian:
BASIC ohne Probleme / Christian Langfelder. —
Haar bei München: Verlag Markt und Technik
(Computer persönlich)
Bd. 2. Übungen: 20 ausgew. Routinen u. Programme
zum Veranschaulichen u. Üben allg. Programmier-
techniken auf CBM-Rechnern (CBM 8032). — 1982. —
ISBN 3-922120-26-1

Die Informationen im vorliegenden Buch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

ISBN 3-922120-26-1
© 1982 by Markt & Technik, 8013 Haar bei München
Alle Rechte vorbehalten
Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Hofmann-Druck, 8900 Augsburg
Printed in Germany

Inhaltsverzeichnis

Einleitung 7
Verwendung der Programmsammlung 7
Flußdiagramme 8

1. Allgemeine Routinen
CURL 11
Textvergleich 17
Rundungs-Routine 21

2. Allgemeine Programme
Kalenderprogramme 27
Vierspezies-Maschine 35
Sortierprogramm 39
Anagramm 45

3. Kommerzielle Programme
Preiskalkulation 51
Fakturierung 57
Optimale Bestellmenge 63
Zinseszins-Rechnung 69
KURWEG 73

4. Statistik
Balkendiagramm 81
Lineare Regression 85

5. Mathematik
Primzahlen 93
Nullstellenbestimmung nach Newton-Raphson 97

6. Spiel- und Lehrprogramme
Wortratespiel 105
Bagels 111
Matrix-Lehrprogramm 117

Einleitung

Dieses Übungsbuch ist die Ergänzung zu dem im gleichen Verlag erschienenen Lehrbuch "BASIC ohne Probleme/Band 1: Unterweisung". Es enthält 20 Routinen und Programme, die vor allem den Anfänger in der Programmierung unterstützen sollen, so rasch wie möglich seinen Rechner erfolgreich zum Einsatz zu bringen. Die Programme setzen sich aus einer Beschreibung, Bedienungsanweisungen, Beispielen, Anmerkungen und reichlich mit Kommentaren versehenen Programmprotokollen zusammen. Die Information wird, sofern sinnvoll und erforderlich, durch Tips, Erläuterungen und Flußdiagramme ergänzt. Die Programme haben in erster Linie die Aufgabe, allgemeine Programmiertechniken zu veranschaulichen. Daher wurde auf besondere, wenn auch gewinnbringende Tricks und Techniken in dieser Sammlung verzichtet.

Die Betonung liegt vielmehr auf den alltäglichen und daher besonders wichtigen Problemen in der Programmierung: die Behandlung und Verarbeitung von Eingabedaten, die Steuerung des Cursors, die Verwendung von Unterprogrammen und die Programmierung von Fehlerausgängen.

Die Programme in diesem Buch sind in sechs Rubriken unterteilt: drei allgemeine Routinen, die auch in anderen Programmen zur Anwendung kommen, fünf allgemeine Programme, fünf kommerziell-technische Anwendungen, zwei Statistikprogramme, zwei Mathematikprogramme und drei (hoffentlich spannende) Lehr- und Spielprogramme. Alle Programme können direkt in einen cbm-Rechner Modell 8032 eingegeben und gestartet werden. Für andere Rechner sind fallweise Anpassungen vorzunehmen.

Verwendung der Programmsammlung

Alle Programme in dieser Sammlung können direkt in einen Rechner vom Typ cbm 8032 eingegeben und ausgeführt werden. Das gilt auch für die Demonstrationsprogramme, die mehr dazu gedacht sind, einen bestimmten Ablauf oder ein besonderes Unterprogramm anschaulich darzustellen. Diese Routinen und Unterprogramme sind in den Programmprotokollen mit Überschriften versehen, so daß sie leicht zu erkennen sind und aus dem eigentlichen Programm zur Verwendung an anderer Stelle herausgelöst werden können.

Alle eigentlichen Programmanweisungen stehen in Programmzeilen, die in der niedrigsten Stelle eine "0" haben. REM-Anweisungen sind immer an erster Stelle in einer Programmzeile zu finden, die in der niedrigsten Stelle eine 1, 2, ... , 9 hat. Wer also vorzieht, seine Programme ohne Kommentare zu schreiben oder auf diese aus Platzgründen verzichten muß, kann die unerwünschten REM-Anweisungen leicht entfernen.

Es sei nochmals darauf hingewiesen, daß dieses Buch sich vorrangig mit Abläufen und Techniken in der Progammierung befaßt. Ein Basic-Handbuch kann und soll dieses Buch daher nicht ersetzen. Eine vollständige Behandlung des cbm-Basic (für Rechner vom Typ 8032) ist in dem Handbuch "BASIC ohne Probleme/Band 1: Unterweisung" zu finden.

Flußdiagramme

Eine übersichtliche grafische Wiedergabe eines Programmablaufs gehört zu jeder vollständigen Dokumentation. Flußdiagramme spielen dabei eine wichtige Rolle, auch wenn sie den Nachteil haben, daß ihre Erstellung bei langen Programmen manchmal recht langwierig ist und der Aufwand dann nicht in einem besonders guten Verhältnis zu dem Nutzen steht.

Für kurze Abläufe gelten beide Einschränkungen nicht. Weil in der Dokumentation der in diesem Buch enthaltenen Programme zuweilen auf Flußdiagramme zurückgegriffen wird, seien an dieser Stelle die wesentlichen Elemente von Flußdiagrammen aufgeführt.

Element	Bedeutung
	Grenzstelle
	Verarbeitung
	Entscheidung/Verzweigung
	Unterprogrammaufruf
	Ein-/Ausgabe
	Connector/Übergangsstelle
	Informationsfluß

1.

Allgemeine Routinen

CURL

In fast jedem Programm, das einen Dialogverkehr zwischen Anwender und Rechner erfordert, wird eine Methode der Cursor-Steuerung benötigt. Das Programm CURL (von CURsor und Löschen) bietet eine effiziente Methode der Cursor-Steuerung und baut gleichzeitig eine Lösch-Zeile auf, da auch das selektive Löschen von Bildschirmteilen in vielen praktischen Anwendungen zum guten Programmkonzept gehört.

CURL ist ein Demonstrationsprogramm, das in dieser Form nicht für den praktischen Einsatz gedacht ist. Es zeigt aber in sehr anschaulicher Weise, wie einfach es ist, eine bestimmte Stelle auf dem Bildschirm anzusteuern und ab dieser Stelle eine vorgegebene Anzahl von Zeichen auf dem Bildschirm zu löschen. Das Programm beginnt damit, die zehn obersten Zeilen auf dem Bildschirm mit einem grafischen Zeichen zu füllen. Mit drei Eingabeparametern werden dann Teile aus diesen horizontalen Balken herausgelöscht.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Ausgabe der obersten zehn Zeilen als durchgehende Balken
4. Eingabeaufforderungen:

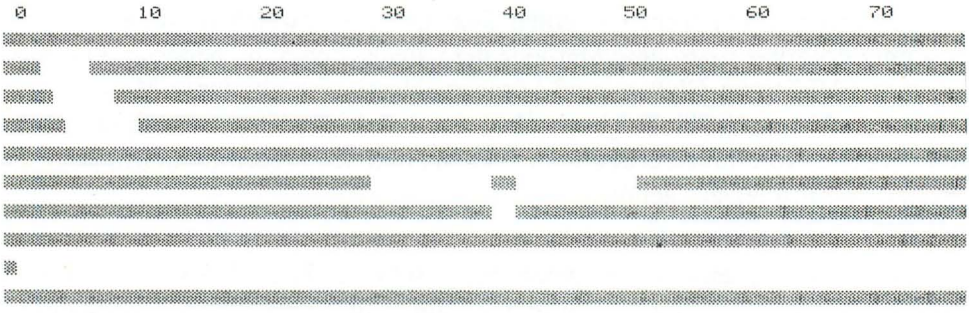
   Reihe, Spalte, Löschzeilen-Länge

   durch entsprechende Eingabe beantworten
5. Löschen des spezifizierten Balkenteils
6. Rückkehr zur nächsten Eingabe in Schritt 4

Anmerkungen

1. Die Eingabe für jede Lösch-Operation wird absichtlich bei jeder weiteren Eingabe um eine Reihe tiefer gesetzt, damit die vorangegangenen Lösch-Operationen protokolliert bleiben.
2. Das Kernstück des Programms bildet eine Routine, die drei Zeichenketten aufbaut, je eine mit 25 Steuercodes für eine Cursor-Bewegung nach unten, eine mit 80 Steuercodes für eine Cursor-Bewegung nach rechts und eine mit 80 Leerzeichen. Durch die Ausgabe von Teilketten kann, relativ zur oberen linken Bildschirmecke, jede Bildschirm-Position erreicht werden.

Beispiel




```
Reihe,Spalte,Loeschzeilen-Laenge? 2,3,4
Reihe,Spalte,Loeschzeilen-Laenge? 3,4,5
Reihe,Spalte,Loeschzeilen-Laenge? 4,5,6
Reihe,Spalte,Loeschzeilen-Laenge? 9,1,78
Reihe,Spalte,Loeschzeilen-Laenge? 6,30,10
Reihe,Spalte,Loeschzeilen-Laenge? 6,42,10
Reihe,Spalte,Loeschzeilen-Laenge? 7,40,2
```

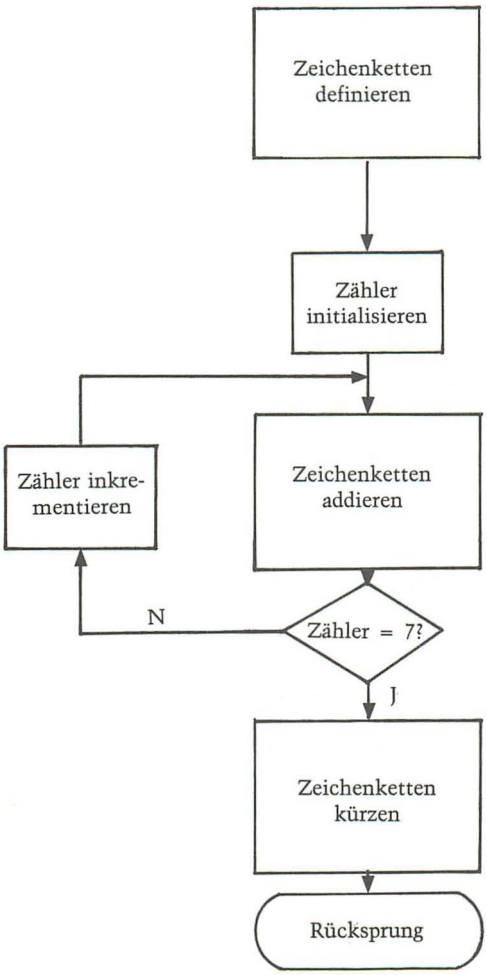
Hinweise

1. Mit 'print chr$(147)' wird der Bildschirm gelöscht (110)
2. Die Zeichenketten 'l$, cu$, cr$' müssen erst mit 'gosub 2110' aufgebaut werden, bevor sie verwendet werden

```
101 rem ---------------------------------------------------------------------
102 rem titel: curl - bildschirmsteuerung und loeschzeile
103 rem ---------------------------------------------------------------------
104 rem
107 rem ---------------------------------------------------------------------
108 rem initialisierung: cursor in linke obere ecke
109 rem ---------------------------------------------------------------------
110 print chr$(147);:r1=11
117 rem ---------------------------------------------------------------------
118 rem unterprogramm-aufruf zur initialisierung der zeichenketten
119 rem ---------------------------------------------------------------------
120 gosub 2110
127 rem ---------------------------------------------------------------------
128 rem horizontal-skala und auffuellen der oberen bildschirm-haelfte
129 rem ---------------------------------------------------------------------
130 for i = 0 to 7: print tab(i*10);:print i*10;:next i:print
140 for j=1 to 10: for i=0 to 79:print chr$(230);:next:next :print
147 rem ---------------------------------------------------------------------
148 rem eingabe: reihe, spalte und laenge der loeschzeile
149 rem ---------------------------------------------------------------------
150 r1=r1+1:r=r1:s=0:gosub 2100:input"Reihe,Spalte,Loeschzeilen-Laenge";r,s,l
157 rem ---------------------------------------------------------------------
158 rem unterprogramm-aufruf fuer die cursor-positionierung; loeschzeile
159 rem ---------------------------------------------------------------------
160 gosub 2100:print left$(l$,l);:goto 150
2097 rem --------------------------------------------------------------------
2098 rem unterprogramm: cursor-steuerung (ueber r1,r,s1,s)
2099 rem --------------------------------------------------------------------
2100 print"▀"left$(cu$,r)left$(cr$,s);:return
2107 rem --------------------------------------------------------------------
2108 rem aufbau der zeichenketten fuer cursor-steuerung und loeschzeile
2109 rem --------------------------------------------------------------------
2110 l$=" ":cu$="▄":cr$="▌":fori=1to7:cu$=cu$+cu$:cr$=cr$+cr$:l$=l$+l$:nexti
2120 cu$=left$(cu$,25):cr$=left$(cr$,80):l$=left$(l$,80):return
2130 end
```

Flußdiagramm 1

Aufbau der Zeichenketten



Zeile	Anmerkungen
2110	␣ = Leerzeichen → = Cursor eine Spalte nach rechts ↓ = Cursor eine Zeile vorrücken
	1$, cr$, cu$ auf 128 Zeichen bringen
2120	1$, cr$, cu$ auf 80, 80 und 25 Zeichen kürzen

Flußdiagramm 2

Cursor-Steuerung	Zeile	Anmerkungen
print "S"	2100	Cursor in linke, obere Ecke
print left$(cu$, r)		Cursor um r Zeilen vorrücken
print left$(cr$, s)		Cursor um s Spalten vorrücken, Wagenrücklauf wird mit "j" unterdrückt
Rücksprung		

Textvergleich

Beim Programmieren kommt es immer wieder vor, daß überprüft werden muß, ob eine Zeichenkette (der Prüfling) in einer anderen (dem Prüfmuster) enthalten ist. Wenn dies der Fall ist, interessiert häufig noch die Information, ab welcher Stelle im Prüfmuster die beiden Zeichenketten übereinstimmen. Als Beispiel sei hier auf die Verwendung von alphanumerischen Zeichen bei der Eingabe von rein numerischen Werten verwiesen. Mit der hier beschriebenen Routine kann in diesem Fall überprüft werden, ob die Eingabe einen Dezimalpunkt enthält und an welcher Stelle er steht.

Das Programm besteht im wesentlichen aus einer Subroutine, die als Unterprogramm in anderen Programmen eingesetzt werden kann. Für Demonstrationszwecke sind einige Texte (Prüfmuster) fest definiert, mit denen der Prüfling (die variable Eingabe) verglichen wird. Bei Übereinstimmung wird die Position im Prüfmuster sowohl optisch als auch numerisch angezeigt.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Eingabeaufforderung:

   Prüfling

   durch entsprechende Eingabe beantworten
4. Ausgabe des Prüfmusters, der Prüfmusternummer und der Position innerhalb des Prüfmusters, ab der eine Übereinstimmung mit dem Prüfling gefunden wurde
5. Rückkehr zur nächsten Eingabe in Schritt 3

Beispiele

Prüfling elle
parzelle 4 5
elle

kellerkind 5 2
elle

Prüfling meran
boomerang 1 4
meran

Prüfling eis
beispiel 2 2
eis

leistung 3 2
eis

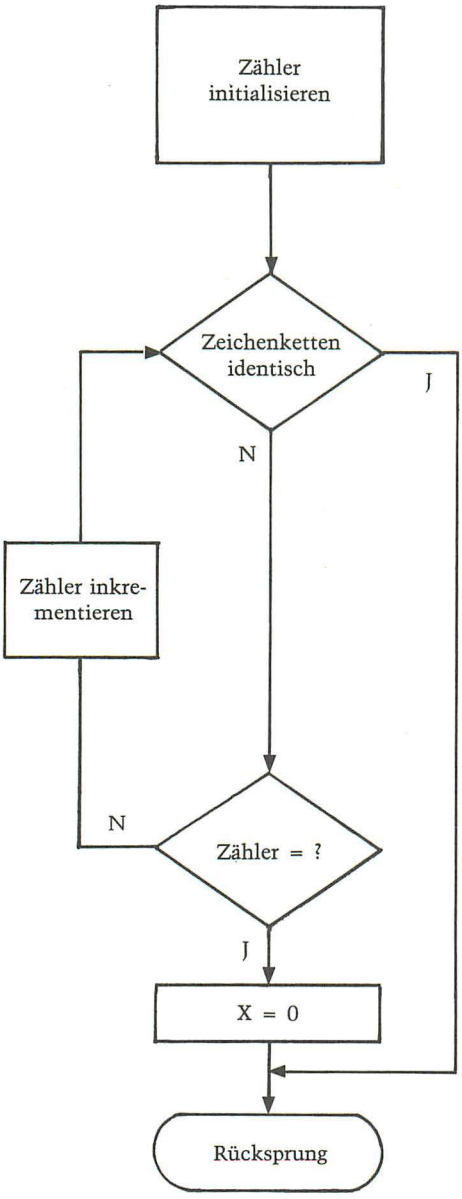
Hinweise

1. Das Unterprogramm, das den Textvergleich vornimmt, schreibt den Wert 'x' zurück. Wenn dieser Null ist, wurde keine Übereinstimmung gefunden (150). Wenn 'x' ungleich Null ist, wird der Wert zur Cursor-Steuerung in 'print tab (x)' hergenommen, um Stammkette und Teilkette untereinander auszudrucken (170).

```
101 rem ------------------------------------------------------------
102 rem titel: textvergleich
103 rem ------------------------------------------------------------
104 rem
107 rem ------------------------------------------------------------
108 rem text-definitionen
109 rem ------------------------------------------------------------
110 a$(1)="boomerang":a$(2)="beispiel":a$(3)=" leistung"
120 a$(4)="parzelle":a$(5)="kellerkind":a$(6)="rangierung"
127 rem ------------------------------------------------------------
128 rem eingabe des prueflings
129 rem ------------------------------------------------------------
130 print:input"Pruefling ";b$:print
140 for i=1to6:a$=a$(i)
147 rem ------------------------------------------------------------
148 rem unterprogramm-aufruf: wenn x=0 dann keine uebereinstimmung
149 rem ------------------------------------------------------------
150 gosub 500:if x=0 then 180
157 rem ------------------------------------------------------------
158 rem ausgabe: pruefling, pruefmuster und position
159 rem ------------------------------------------------------------
160 print a$;i;x
170 printtab(x-1);b$:print
180 nexti:goto 130
496 rem ------------------------------------------------------------
497 rem unterprogramm zur ueberpruefung, ob eine teilkette b$
498 rem vollstaendig in einer zeichenkette a$ enthalten ist
499 rem ------------------------------------------------------------
500 for x=1 to len(a$)-len(b$)+1
510 if b$=mid$((a$),x,len(b$)) then return
520 next x:x=0: return
530 end
```

Flußdiagramm

Vergleich von a$ und b$



Zeile	Anmerkungen
500	Der Textvergleich beginnt mit der ersten Stelle des Prüfmusters
510	Abfrage, ob Prüfling b$ über seine volle Länge (len(b$)) in Prüfmuster a$ ab Position x enthalten ist
520	Abfrage, ob x gleich y (y = len(b$)-len(a$) + 1
	Keine Übereinstimmung von Prüfling und Prüfmuster

Rundungs-Routine

Es ist häufig wünschenswert, ein numerisches Ergebnis nicht mit voller (vielleicht neunstelliger) Genauigkeit zu verwenden, sondern die signifikanten Ziffern auf eine bestimmte Anzahl zu begrenzen. Dieser mit Rundung bezeichnete Prozeß verbessert nicht nur die Optik einer Ausgabe, sondern wird auch dann benötigt, wenn es gilt, eine Zahl vor ihrer weiteren Verwendung in eine entsprechende Form zu bringen. Das ist dann beispielsweise der Fall, wenn ein Ergebniswert als Index, also als ganzzahliger Wert, verwendet wird. Hier genügt bekanntlich die INT-Funktion nicht, weil diese lediglich den Dezimalteil einer Zahl abtrennt, ohne ihn vorher hinsichtlich seines Beitrags zu überprüfen.

Zur Rundung wird die Formel

```
X = INT(Y★10↑Z + 0.5)/10↑Z
```

verwendet, wobei

Y = die zu rundende Zahl
Z = die Position der letzten signifikanten Ziffer
(relativ zum Dezimalpunkt)
X = das Ergebnis

bedeuten.

Die Wirkung der Formel läßt sich am besten anhand eines Beispiels verfolgen.

Als Eingabe seien die Werte

Y = 456.617
Z = 0

angenommen.

Schrittweise ergibt sich folgender Ablauf

```
INT(456.617★1 + 0.5)/1    (10↑0 = 1)
INT(456.617   + 0.5)/1
INT(457.117)/1
457/1
457
```

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Eingabeaufforderungen:

   Zu rundende Zahl:

   Anzahl Stellen:

   durch entsprechende Eingaben beantworten
4. Ausdruck der Ergebniswerte
5. Rückkehr zur nächsten Eingabe

Anmerkungen

1. Die Rundungs-Routine erfüllt als selbständiges Programm nur einen Demonstrationszweck. Die Routine ist vielmehr als Unterprogramm gedacht. Das Unterprogramm wird mit der zu rundenden Zahl und der gewünschten Anzahl von signifikanten Stellen versorgt. Wenn, wie bei DM-Beträgen, immer in der zweiten Stelle nach dem Komma gerundet werden soll, kann dieser Parameter (im Programm mit 'z' angegeben) als Konstante eingesetzt werden.
2. Die Routine eignet sich auch für den Fall, daß mehrere Daten (allerdings dann nicht mit voller Genauigkeit) in einem Datenwort gepackt werden sollen. Erst werden die Zahlen auf die gewünschte Länge gebracht, dann mit 10 oder einem Vielfachen von 10 multipliziert und zu einer zweiten zu speichernden Zahl addiert. Bei der Entschlüsselung wird der Ablauf umgekehrt.

Beispiele

Zu rundende Zahl: ? 456.617
Anzahl Stellen: ? 2

456.62 456.617

Zu rundende Zahl: ? 456.617
Anzahl Stellen: ? 1

456.6 456.617

Zu rundende Zahl: ? 456.617
Anzahl Stellen: ? 0

457 456.617

Zu rundende Zahl: ? 456.617
Anzahl Stellen: ? —2

500 456.617

```
1001 rem ------------------------------------------------------------------
1002 rem titel: rundungs-routine
1003 rem ------------------------------------------------------------------
1004 rem
1007 rem ------------------------------------------------------------------
1008 rem loeschen des bildschirms und eingabeteil
1009 rem ------------------------------------------------------------------
1010 print chr$(147)
1020 input"Zu rundende Zahl: ";y
1030 input"Anzahl Stellen  : ";z
1040 x= int(y*10↑z + 0.5)/(10↑z)
1047 rem ------------------------------------------------------------------
1048 rem ausgabe der gerundeten und der urspruenglichen zahl
1049 rem ------------------------------------------------------------------
1050 print:printx,y:print
1060 goto 1020
1070 end
ready.
```

2.

Allgemeine Programme

Kalenderprogramme

Zwei Unterprogramme, die zusammen oder auch getrennt verwendet werden dürfen, erlauben die Umwandlung eines Datums in eine Julianische Tageszahl und ihre Rückwandlung, die Berechnung der Anzahl von Tagen zwischen zwei Daten oder eines zukünftigen Datums, wenn ein Anfangsdatum und die Anzahl dazwischenliegender Tage bekannt ist, und die Ermittlung des Wochentags zu einem gegebenen Datum.

Das Programm überprüft, ob das eingegebene Datum ein zulässiges Datum darstellt. So werden nicht nur der 31. Juni 1982 oder 29. Februar 1983 abgelehnt, sondern auch das Eingabeformat muß genau eingehalten werden. Als Begrenzer zwischen den Zifferngruppen sind alle nichtnumerischen Zeichen zugelassen. Bei der Ausgabe wird als Begrenzer immer ein Punkt verwendet.

Bei der Eingabe des Datums muß das Format 'tt.mm.jj' eingehalten werden. Den Umwandlungsroutinen liegen folgende Formeln zugrunde:

A. Berechnung einer Julianischen Tageszahl

JTZ = INT(365.25★JJ) + INT(30.6001★MM) + tt

mit den Bedingungen

JJ = jj wenn mm > 2
= jj —1 wenn mm = 1 oder mm = 2

MM = mm wenn mm > 2
= mm + 13 wenn mm = 1 oder mm = 2

wobei

JTZ Julianische Tageszahl
tt Tag
mm Monat
jj Jahr (ohne Angabe des Jahrhunderts)

bedeuten.

B. Berechnung eines Tagesdatums zu einer gegebenen JTZ

JJ = INT((JTZ — 122.1)/365.25)

MM = INT((JTZ — INT(365.25★JJ))/30.6001)

tt = JTZ — INT(365.22★JJ) — INT(30.6001★MM)

mit den Bedingungen

mm = MM — 13 wenn MM = 14 oder MM = 15
= MM — 1 wenn MM < 14

jj = JJ wenn mm > 2
= JJ + 1 wenn mm = 1 oder 2 oder mm = 2

Für die Berechnung des Wochentags zu einem gegebenen Tagesdatum gilt die Beziehung

Wochentag (als Zahl im Bereich 0 bis 6) = 7★(JTZ/7 — INT(JTZ/7))

Bedienungsanweisung (Kalenderprogramm 1)

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Eingabeaufforderungen:

   1. Datum
   2. Datum
   Anzahl Tage

   durch entsprechende Eingabe beantworten
4. Ausdruck des Ergebnisses
5. Rückkehr zu Schritt 3 für die nächste Eingabe

Bedienungsanweisung (Kalenderprogramm 2)

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Eingabeaufforderungen:

   1. Datum

   durch entsprechende Eingabe beantworten
4. Ausdruck des Ergebnisses
5. Rückkehr zu Schritt 3 für die nächste Eingabe

Beispiel (Kalenderprogramm 1)

Das Kalenderprogramm 1 löst die Gleichung

Datum1 — Datum2 = Anzahl Tage

Das Datum wird in dem Format tt.mm.jj eingegeben. Die Eingabeaufforderung für den gesuchten Wert wird mit der Taste RETURN (leere Eingabe) quittiert.

1. Datum
23.05.82

2. Datum

Anzahl Tage
1000

23.05.82 — 1000 — 16.02.85

1. Datum
23.05.82

2. Datum
23.05.83

Anzahl Tage

23.05.82 — 365 — 23.05.83

Beispiel (Kalenderprogramm 2)

Das Kalenderprogramm 2 ermittelt zu einem in dem Format tt.mm.jj eingegebenen Datum den zugehörigen Wochentag.

1. Datum
26.05.82

Mittwoch

1. Datum
25.12.82

Sonnabend

Anmerkungen

1. Das Format 'tt.mm.jj' soll die Eingabe des Datums, besonders bei kommerziellen Anwendungen, erleichtern. Der Algorithmus arbeitet mit der vollen Jahreszahl.
2. Nach der Rückkehr aus dem Unterprogramm 1 (Berechnung der JTZ) wird in einer Variablen vermerkt, ob bei der Eingabe die Bedingungen für ein zulässiges Datum eingehalten wurden.

Hinweise

1. Wenn von drei möglichen Eingaben maximal zwei gleichzeitig zugelassen sind, muß vermerkt werden, welche Eingaben getätigt wurden (180-200). Je nach Eingabe nimmt 'p' die Werte 1, 2 oder 3 an, der dann für eine verteilte Verzweigung verwendet werden kann.
2. Die Anweisung 'p = 3—p' (270) für p = 1 oder 2 wandelt den Wert p in 2 bzw. 1 um.

```
101 rem ----------------------------------------------------------------------
102 rem titel: kalenderprogramm 1
103 rem ----------------------------------------------------------------------
104 rem
107 rem ----------------------------------------------------------------------
108 rem initialisierung: open 1,0 fuer input#: programm-kurzbeschreibung
109 rem ----------------------------------------------------------------------
110 print chr$(147): open 1,0
120 print"Das Kalenderprogramm 1 loest die Gleichung":print
130 print"Datum1 - Datum2 = Anzahl Tage":print
140 print"Das Datum wird im dem Format tt.mm.jj eingegeben. Die"
150 print"Eingabeaufforderung fuer den gesuchten Wert wird mit"
160 print"der Taste RETURN (leere Eingabe) quittiert"
170 print:print:p=0:q=-1
177 rem ----------------------------------------------------------------------
178 rem eingabeteil: bei richtiger eingabe nimmt p werte von 1, 2 oder 4 an
179 rem ----------------------------------------------------------------------
180 print"1. Datum":input#1,x$(1):print:print:if len(x$(1))=0 then p=p+2
190 print"2. Datum":input#1,x$(2):print:print:if len(x$(2))=0 then p=p+1:q=1
200 print"Anzahl Tage":input#1,x$(3):print:print:if len(x$(3))=0 then p=p+4
207 rem ----------------------------------------------------------------------
208 rem verteilte verzweigung: fehlermeldung bei unzulaessiger eingabe
209 rem ----------------------------------------------------------------------
210 on p goto 240, 240, 220, 280
220 print"Falsche Eingaben": goto 170
227 rem ----------------------------------------------------------------------
228 rem p = 1 oder 2: gesucht wird datum1 oder datum2: q = 1 oder -1
229 rem ----------------------------------------------------------------------
240 if len(x$(p))<>8 then goto 320
250 q1$=x$(p):gosub 1100:if q0(0)=0 then 310
260 q0(4)=q0(4)+val(x$(3))*q:gosub 1300
270 let p=3-p: x$(p)= q1$: goto 320
277 rem ----------------------------------------------------------------------
278 rem p = 4: gesucht wird anzahl tage zwischen zwei daten
279 rem ----------------------------------------------------------------------
280 if len(x$(1))<> 8 or len(x$(2))<>8 then 320
290 q1$=x$(1):gosub 1100:q=q0(4):if q0(0)=0 then 320
300 q1$=x$(2):gosub 1100:x$(3)=str$(q0(4)-q):if q0(0)=0 then 320
310 goto 330
317 rem ----------------------------------------------------------------------
318 rem fehlermeldung: datum-format tt.mm.jj nicht eingehalten
319 rem ----------------------------------------------------------------------
320 print"Falsches Format": goto 170
327 rem ----------------------------------------------------------------------
328 rem ausgabe-anweisung: ruecksprung-anweisung fuer naechstes beispiel
329 rem ----------------------------------------------------------------------
330 print x$(1)" - "x$(3)" - "x$(2):print:goto 170
340 end
```

```
1100 for i=1to3:letq0(i)=val(mid$(q1$,(i-1)*3+1,2)):nexti
1110 q0(0)=1:fori=1to8:q0(4)=int((asc(mid$(q1$,i,1))+2)/10)-4
1120 if(i=3ori=6)andq0(4)=1thenq0(0)=0
1130 if(i<>3andi<>6)andq0(4)<>1thenq0(0)=0
1140 nexti:i=0:ifint(q0(3)/4)=q0(3)/4theni=1
1150 ifq0(2)=2andq0(1)>28+ithenq0(0)=0
1160 ifq0(1)>31orq0(2)<1orq0(2)>12orq0(3)<78thenq0(0)=0
1170 if(q0(2)=4orq0(2)=6)andq0(1)>30thenq0(0)=0
1180 if(q0(2)=9orq0(2)=11)andq0(1)>30thenq0(0)=0
1190 ifq0(2)<>1andq0(2)<>2then1210
1200 q0(3)=q0(3)-1:q0(2)=q0(2)+12
1210 q0(3)=q0(3)+1900:q0(2)=q0(2)+1
1220 q0(4)=int(365.25*q0(3))+int(30.6001*q0(2))+q0(1):return
1300 q0(3)=int((q0(4)-122.1)/365.25)
1310 q0(2)=int((q0(4)-int(q0(3)*365.25))/30.6001)
1320 q0(1)=q0(4)-int(q0(3)*365.25)-int(q0(2)*30.6001)
1330 ifq0(2)=14orq0(2)=15thenq0(2)=q0(2)-12
1340 q0(2)=q0(2)-1:ifq0(2)=1orq0(2)=2thenq0(3)=q0(3)+1
1350 q1$="":ifq0(1)<10thenq1$="0"
1360 q1$=q1$+mid$(str$(q0(1)),2)+".":ifq0(2)<10thenq1$=q1$+"0"
1370 q1$=q1$+mid$(str$(q0(2)),2)+"."+mid$(str$(q0(3)-1900),2):return
ready.
```

```
101 rem ----------------------------------------------------------------------
102 rem kalenderprogramm 2
103 rem ----------------------------------------------------------------------
104 rem
107 rem ----------------------------------------------------------------------
108 rem initialisierung: open 1,0 fuer input#: programm-kurzbeschreibung
109 rem ----------------------------------------------------------------------
110 t$(0)="Freitag":t$(1)="Sonnabend":t$(2)="Sonntag":t$(3)="Montag"
120 t$(4)="Dienstag":t$(5)="Mittwoch":t$(6)="Donnerstag"
130 print chr$(147): open 1,0
140 print"Das Kalenderprogramm 2 ermitttelt zu einem in dem Format"
150 print"tt.mm.jj eingegebenen Datum den zugehoerigen Wochentag"print:print
157 rem ----------------------------------------------------------------------
158 rem eingabe und ueberpruefung des datums
159 rem ----------------------------------------------------------------------
160 print"1. Datum":input#1,x$(1):print:print
170 if len(x$(1))<>8 then goto 210
180 q1$=x$(1):gosub 1100:if q0(0)=0 then 210
187 rem ----------------------------------------------------------------------
188 rem berechnung des wochentages als zahl zwischen 0 und 6
189 rem ----------------------------------------------------------------------
190 p=q0(4)/7: q=int(((p-int(p))*7)+.5))
200 print t$(q):print goto 160
207 rem ----------------------------------------------------------------------
208 rem fehlermeldung: datum-format tt.mm.jj nicht eingehalten
209 rem ----------------------------------------------------------------------
210 print"Falsches Format": goto 160
220 end
```

```
1001 rem ----------------------------------------------------------------------
1002 rem umwandlung eines tagesdatums in eine julianische tageszahl und
1003 rem rueckwandlung einer julianischen tageszahl in ein tagesdatum
1004 rem ----------------------------------------------------------------------
1005 rem
1097 rem ----------------------------------------------------------------------
1098 rem umwandlung eines datums (tt.mm.jj) in eine julianische tageszahl
1099 rem ----------------------------------------------------------------------
1100 fori=1to3:letq0(i)=val(mid$(q1$,(i-1)*3+1,2)):nexti
1110 q0(0)=1:fori=1to8:q0(4)=int((asc(mid$(q1$,i,1))+2)/10)-4
1120 if(i=3ori=6)andq0(4)=1thenq0(0)=0
1130 if(i<>3andi<>6)andq0(4)<>1thenq0(0)=0
1140 nexti:i=0:ifint(q0(3)/4)=q0(3)/4theni=1
1150 ifq0(2)=2andq0(1)>28+ithenq0(0)=0
1160 ifq0(1)>31orq0(2)<1orq0(2)>12orq0(3)<78thenq0(0)=0
1170 if(q0(2)=4orq0(2)=6)andq0(1)>30thenq0(0)=0
1180 if(q0(2)=9orq0(2)=11)andq0(1)>30thenq0(0)=0
1190 ifq0(2)<>1andq0(2)<>2then1210
1200 q0(3)=q0(3)-1:q0(2)=q0(2)+12
1210 q0(3)=q0(3)+1900:q0(2)=q0(2)+1
1220 q0(4)=int(365.25*q0(3))+int(30.6001*q0(2))+q0(1):return
1297 rem ----------------------------------------------------------------------
1298 rem umwandlung einer julianischen tageszahl in ein datum (tt.mm.jj)
1299 rem ----------------------------------------------------------------------
1300 q0(3)=int((q0(4)-122.1)/365.25)
1310 q0(2)=int((q0(4)-int(q0(3)*365.25))/30.6001)
1320 q0(1)=q0(4)-int(q0(3)*365.25)-int(q0(2)*30.6001)
1330 ifq0(2)=14orq0(2)=15thenq0(2)=q0(2)-12
1340 q0(2)=q0(2)-1:ifq0(2)=1orq0(2)=2thenq0(3)=q0(3)+1
1350 q1$="":ifq0(1)<10thenq1$="0"
1360 q1$=q1$+mid$(str$(q0(1)),2)+".":ifq0(2)<10thenq1$=q1$+"0"
1370 q1$=q1$+mid$(str$(q0(2)),2)+"."+mid$(str$(q0(3)-1900),2):return
1380 end
```

Vierspezies-Maschine

Es gibt keinen Grund, einen leistungsfähigen Rechner nicht auch für "niedrigere" Arbeiten einzusetzen, sofern sich dabei Vorteile gegenüber herkömmlichen Methoden ergeben.

Der Rechner als Vierspezies-Maschine ist eine typische kleine Anwendung. Der Anwender gibt einen Wert ein, der gemäß dem eingegebenen Operator (+/—, ★, :) mit einem Faktor verknüpft wird. Das entspricht einer Lösung der Gleichung

$y = a \star x + b$

wobei

x	Eingabewert
y	Endergebnis
a = 1, b = Faktor	bei Addition und Subtraktion
b = 0, a = Faktor	bei Multiplikation
b = 0, a = 1/Faktor	bei Division

darstellen.

Das Endergebnis kann noch auf Wunsch gerundet werden.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Eingabeaufforderungen:

   Faktor
   Operator
   Rundung

   durch entsprechende Eingaben beantworten
4. Die Eingabeaufforderung:
   Eingabewert
   durch eine entsprechende Eingabe beantworten bzw. die Serie durch RETURN (leere Eingabe) beenden
5. Rückkehr zur nächsten Serie in Schritt 3

Anmerkungen

1. Das Programm verwendet die Eigenschaft der INPUT # -Anweisung, daß durch eine leere Eingabe das Programm nicht auf Ende läuft, sondern normal fortgesetzt wird, sofern sich der Cursor an der Anfangsposition einer Zeile befindet.
2. Für die Rundung des Ergebniswertes wird die im Programm "Rundungs-Routine" beschriebene Konstruktion verwendet.

Beispiel

Dieses Programm erlaubt die Verknüpfung eines Eingabewertes mit einer Konstanten. Die Art der Verknüpfung wird durch die Wahl des Operators (+ /— ★ :) bestimmt, wobei "1" für "+/—", "2" für "★" und "3" für ":" verwendet wird. Die Anzahl der signifikanten Stellen im Endergebnis wird mit einem Parameter gesteuert. Durch Drücken der Taste RETURN ohne Eingabewert wird eine Serie beendet.

Faktor 12	Operator 2	Rundung 1
Eingabewert		
14	170	
56.789	680	
1.111	10	
21.21	250	

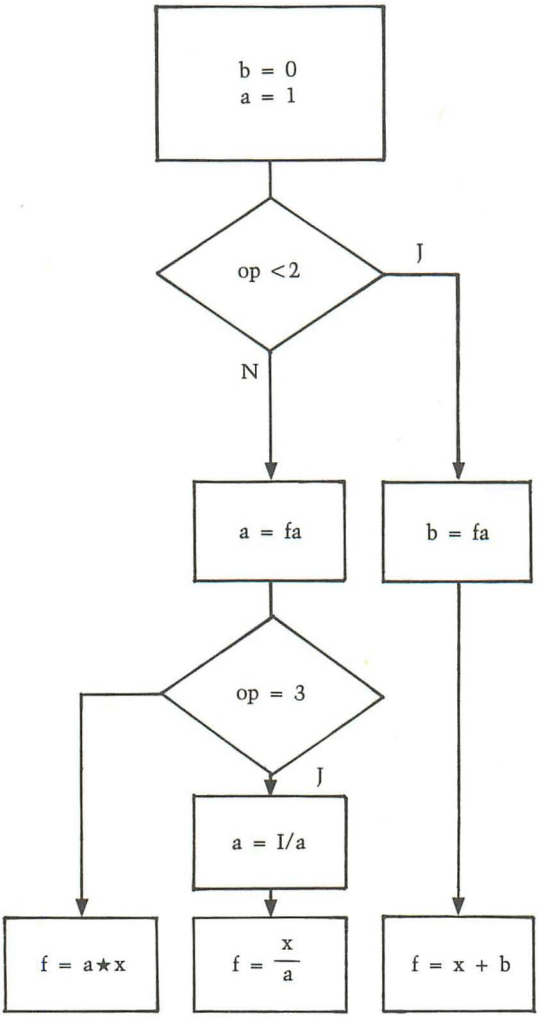
Faktor 10	Operator 3	Rundung 2
Eingabewert		
123.456	12.35	
23.007	2.3	
12.09	1.21	
99	9.9	

Faktor 20	Operator 1	Rundung 0
Eingabewert		
12.89	33	
12.45	32	
—34.749	—15	

```
101 rem ---------------------------------------------------------------------
102 rem vierspezies-rechner mit konstantem faktor und waehlbarer rundung
103 rem ---------------------------------------------------------------------
104 rem
107 rem ---------------------------------------------------------------------
108 rem initialisierung: open 1,0 fuer input#: programm-kurzbeschreibung
109 rem ---------------------------------------------------------------------
110 print chr$(147): open 1,0:print
121 print"Dieses Programm erlaubt die Verknuepfung eines Eingabewertes"
122 print"mit einer Konstanten. Die Art der Verknuepfung wird durch die"
123 print"Wahl des Operators (+/- * :) bestimmt, wobei '1' fuer '+/-',"
124 print"'2' fuer '*' und '3' fuer ':' verwendet wird. Die Anzahl der"
125 print"der signifikanten Stellen im Endergebnis wird mit einem Para-"
126 print"meter gesteuert. Durch Druecken der Taste RETURN ohne Eingabe-"
127 print"wert wird eine Serie beendet"
167 rem ---------------------------------------------------------------------
168 rem eingabeteil : fa=faktor, op=operator, ru=rundung
169 rem ---------------------------------------------------------------------
170 print:print"Faktor ";:input#1,fa:print spc(2);
180 print"Operator ";:input#1,op:print spc(2);
190 print"Rundung ";:input#1,ru:print:print
197 rem ---------------------------------------------------------------------
198 rem aufbau des ausdrucks   a * x + b
199 rem ---------------------------------------------------------------------
200 b=0:a=1:if op <2 then b=fa:goto240
210 a=fa:if op=3 then a=1/a
237 rem ---------------------------------------------------------------------
238 rem eingabeteil fuer die variable x; wenn x=0 dann ende der serie
239 rem ---------------------------------------------------------------------
240 print"Eingabewert"
250 input#1,x:if x=0 then 170
267 rem ---------------------------------------------------------------------
268 rem berechnung und rundung von   a * x + b
269 rem ---------------------------------------------------------------------
270 y=int((a*x+b)*10↑ru+.5)/int(10↑ru)
277 rem ---------------------------------------------------------------------
278 rem berechnung der druckposition und ausdruck des endergebnisses
279 rem ---------------------------------------------------------------------
280 print tab(22-len(str$(int(y)*10)));y
290 goto 250
300 end
```

Flußdiagramm

Aufbau von f = a★x+b



Zeile	Anmerkungen
200	Voreinstellung von a und b
	Bei 'op' kleiner als 2 wird b gleich 'fa' gesetzt
210	a wird gleich 'fa' gesetzt
	Bei 'op' gleich 3 wird a gleich 1/a gesetzt

Sortierprogramm

Für einige Aufgaben ist der Rechner durch Eigenschaften wie hohe Arbeitsgeschwindigkeit und die Handhabung iterativer Abläufe geradezu prädestiniert. Zu diesen Aufgaben zählt auch das Sortieren von Information, was sich schon an der großen Vielfalt von Sortierprogrammen erkennen läßt.

Das hier vorgestellte Insertion-Sortierprogramm stellt für Datenmengen von bis zu 20 Elementen ein optimales Verfahren dar. Es läßt sich natürlich auch für größere Datenmengen verwenden, wenn die Sortierzeit nicht von ausschlaggebender Wichtigkeit ist.

Die Insertion-Methode geht von einer bereits sortierten Sequenz aus, in die ein weiteres Element eingefügt wird. Eine aus n Elementen bestehende Datenmenge ist also nach maximal n-1 Einfügungen sortiert.

Bedienungsanweisung (Sortierprogramm 1 — numerische Daten)

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Eingabeaufforderung:

   Anzahl Werte (n > 1)

   durch entsprechende Eingabe beantworten
4. Ausgabe der unsortierten Datenelemente
5. Ausgabe der sortierten Datenelemente
6. Anzeige der Sortierzeit (in Sekunden)
7. Rückkehr zur nächsten Eingabe in Schritt 3

Bedienungsanweisung (Sortierprogramm 2 — alphanumerische Daten)

Mit Ausnahme von Schritt 4 ist der Programmablauf mit dem von Sortierprogramm 1 identisch. Dieser lautet hier

4. Eingabe der zu sortierenden Zeichenketten

Anmerkungen

1. Für das Sortieren von numerischen Daten und alphanumerischen Größen liegen zwei verschiedene Fassungen des Programms vor. Während Zeichenketten nur als solche behandelt werden können, ist es durchaus zulässig, Zahlen als Zeichenfolgen einzugeben und diese dann mit dem Sortierprogramm für Zeichenketten zu sortieren. Gegebenenfalls müssen die Zahlen vorher mit der STR$-Funktion erst in alphanumerische Werte umgewandelt werden.
2. Die in den numerischen Beispielen verwendeten Daten wurden mit dem Zufallsgenerator des Rechners erzeugt. Um die Ausgabe auf dem Bildschirm so übersichtlich wie möglich zu gestalten, wurden die Zahlen auf den Einserbereich beschränkt.
3. Um die Sortierdauer zu messen, wurde die TI-Funktion des Rechners vor und nach dem Sortiervorgang aufgerufen. Damit läßt sich ganz deutlich das nichtlineare Verhältnis zwischen der Anzahl der zu sortierenden Elemente und der Sortierzeit demonstrieren.

Beispiele (Sortierprogramm 1)

Anzahl Werte (n > 1) ? 20

9 9 8 4 9 1 4 2 4 5 0 6 1 9 8 0 3 6 0 3
0 0 0 1 1 2 3 3 4 4 4 5 6 6 8 8 9 9 9 9

Sortierzeit 2.6 Sekunden

Anzahl Werte (n > 1) ? 25

6 3 3 7 3 6 5 4 6 3 4 2 6 3 3 6 3 0 9 9 9 8 7 3 7
0 2 3 3 3 3 3 3 3 3 4 4 5 6 6 6 6 6 7 7 7 8 9 9 9

Sortierzeit 4.1 Sekunden

Anzahl Werte (n > 1) ? 10

1 6 7 2 3 6 1 1 1 9
1 1 1 1 2 3 6 6 7 9

Sortierzeit .733333333 Sekunden

Beispiel (Sortierprogramm 2)

```
Anzahl Werte (n > 1) ? 7

? peter
? paul
? Peter
? Paul
? Quitte
? Zitter
? ende

ende
paul
peter
Paul
Peter
Quitte
Zitter

Sortierzeit 0.4 Sekunden

Anzahl Werte (n > 1) ?
```

Hinweise

1. Mit der Anweisung 'dim a(n)' (130) wird die Größe des Feldes a dynamisch festgelegt. Damit diese Anweisung wiederholt ausgeführt werden kann, muß das Programm jedesmal mit einer RUN-Anweisung gestartet werden (160).

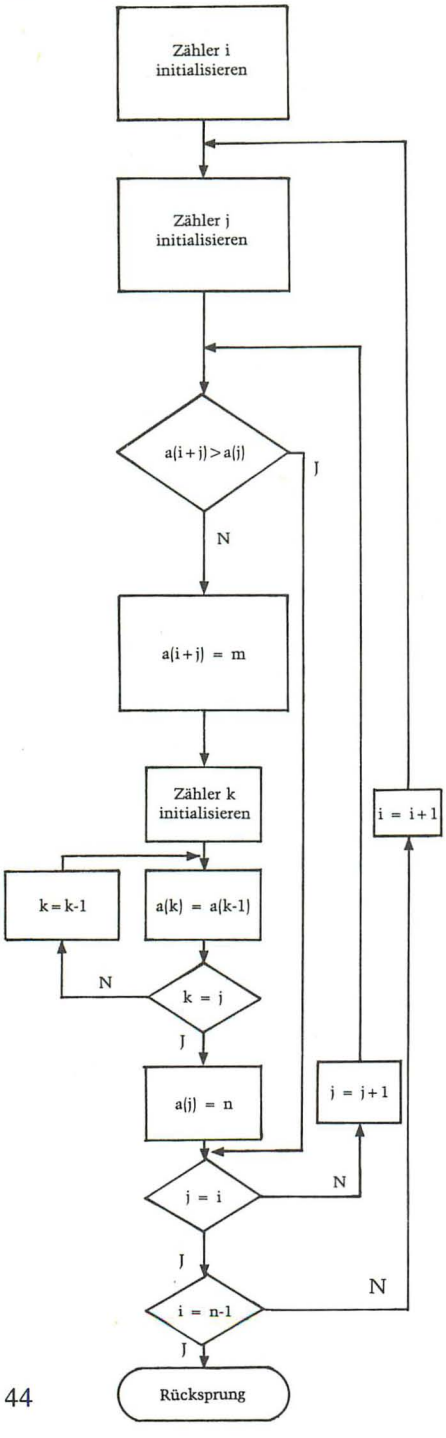
2. Die Variable 'ti' (150) ist eine Systemvariable, in der das System die verstrichene Zeit nach Einschalten des Geräts absetzt.

```
101 rem ----------------------------------------------------------------
102 rem titel: sortierprogramm 1 (insertion-sortierung)
103 rem ----------------------------------------------------------------
104 rem
107 rem ----------------------------------------------------------------
108 rem intialisierung
109 rem ----------------------------------------------------------------
110 print chr$(147)
117 rem ----------------------------------------------------------------
118 rem eingabe der feld-groesse n
119 rem ----------------------------------------------------------------
120 print:input"Anzahl Werte ( n > 1 ) ";n:print
130 dim a(n)
137 rem ----------------------------------------------------------------
138 rem erzeugung von n pseudo-zufallszahlen
139 rem ----------------------------------------------------------------
140 fori=1 to n:a(i)=int(rnd(0)*10):nexti:gosub 500
147 rem ----------------------------------------------------------------
148 rem zeitnehmung und unterprogramm-aufrufe
149 rem ----------------------------------------------------------------
150 let t=ti: gosub 1100: t=ti-t: gosub 500
160 print"Sortierzeit "t/60" Sekunden": run 120
497 rem ----------------------------------------------------------------
498 rem unterprogramm: ausgabe der feldelemente a(n)
499 rem ----------------------------------------------------------------
500 forl=1ton:printa(l);:nextl:print:return
1097 rem ---------------------------------------------------------------
1098 rem unterprogramm: insertion-sortierung der elemente von a(n)
1099 rem ---------------------------------------------------------------
1100 for i=1to n-1
1110 for j=1 to i
1120 if a(i+1)>=a(j)then1140
1130 m=a(i+1):fork=i+1tojstep-1:a(k)=a(k-1):nextk:a(j)=m:j=i
1140 next j:nexti:return
1150 end
```

```
101 rem ----------------------------------------------------------------
102 rem titel: sortierprogramm 2 (insertion-sortierung)
103 rem ----------------------------------------------------------------
104 rem
107 rem ----------------------------------------------------------------
108 rem intialisierung
109 rem ----------------------------------------------------------------
110 print chr$(147)
117 rem ----------------------------------------------------------------
118 rem eingabe der feld-groesse n
119 rem ----------------------------------------------------------------
120 print:input"Anzahl Werte ( n > 1 ) ";n:print
130 dim a$(n)
137 rem ----------------------------------------------------------------
138 rem eingabe der zeichen-ketten
139 rem ----------------------------------------------------------------
140 for i=1 to n: input a$(i):nexti
147 rem ----------------------------------------------------------------
148 rem zeitnehmung und unterprogramm-aufrufe
149 rem ----------------------------------------------------------------
150 let t=ti: gosub 1100: t=ti-t: gosub 500
160 print"Sortierzeit "t/60" Sekunden": run 120
497 rem ----------------------------------------------------------------
498 rem unterprogramm: ausgabe der feldelemente a(n)
499 rem ----------------------------------------------------------------
500 print:for l=1 to n:printa$(l):nextl:print:return
1097 rem ---------------------------------------------------------------
1098 rem unterprogramm: insertion-sortierung der elemente von a$(n)
1099 rem ---------------------------------------------------------------
1100 for i=1to n-1
1110 for j=1 to i
1120 if a$(i+1)>=a$(j)then1140
1130 m$=a$(i+1):fork=i+1tojstep-1:a$(k)=a$(k-1):nextk:a$(j)=m$:j=i
1140 next j:nexti:return
1150 end
```

Flußdiagramm

Insertion-Routine



Zeile	Anmerkungen
1100	Zu sortieren sind n Felder
1110	Ein neues Element wird mit den bereits sortierten Elementen verglichen
1120	Ein neues Element wird nicht eingefügt, wenn es größer als die bereits sortierten Elemente ist
1130	Das neue Element wird zwischengespeichert
	Das sortierte Feld wird ab dem j-ten Element um eine Position verschoben
	Das zwischengespeicherte Element wird in den sortierten Bereich eingefügt
1140	Ende der j- und i- Schleifen

Anagramm

Die Wortbildung durch Buchstabenversetzung eignet sich nicht nur für ungeduldige Kreuzworträtseltüftler. Eine ganze Reihe von Applikationen erfordert die Umbildung der einzelnen Elemente einer Zeichenkette, um die verschiedenen Kombinationen auf gewisse Eigenschaften zu überprüfen.

Die Eingaben für die Routine bestehen aus der Länge der Stamm-Zeichenkette (N), der Länge der zu bildenden Teilkette (K) und der alphanumerischen Zeichenfolge der Stamm-Zeichenkette. Die in Register R(1), ..., R(K) enthaltene Variation der Zahlen N wird mit der Routine durch die nächste Variation ersetzt.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Eingabeaufforderungen:

   N? — Länge der Stamm-Zeichenkette
   K? — Länge der Teil-Zeichenkette
   Kette? — Zeichenkette

   durch entsprechende Eingabe beantworten
4. Ausgabe der Variation
5. Rückkehr zur nächsten Eingabe in Schritt 3

Beispiel 1

N? 5
K? 3
Kette? 12340

123 124 120 132 134 130 142 143 140 102 103
104 213 214 210 231 234 230 241 243 240 201
203 204 312 314 310 321 324 320 341 342 340
301 302 304 412 413 410 421 423 420 431 432
430 401 402 403 012 013 014 021 023 024 031
032 034 041 042 043

Beispiel 2

N? 4
K? 4
Kette? beil

beil beli biel bile blei blie ebil ebli eibl eilb elbi elib ibel
ible iebl ielb ilbe ileb lbei lbie lebi leib libe lieb

N?

Hinweise

1. Die Länge der Zeichenkette wird durch die Anweisung 'dim a(20)' (110) festgelegt. Dieser Wert kann nach Bedarf geändert werden.

```
101 rem ---------------------------------------------------------------------
102 rem titel: anagramm
103 rem ---------------------------------------------------------------------
104 rem
107 rem ---------------------------------------------------------------------
108 rem intialisiserung: feldvereinbarung
109 rem ---------------------------------------------------------------------
110 dim a(20):print chr$(147)
117 rem ---------------------------------------------------------------------
118 rem eingabe: anzahl elemente, anzahl subelemente, zeichenkette
119 rem ---------------------------------------------------------------------
120 print:print:input"N";n
130 input"K";k:fori=1tok:a(i)=i:nexti
140 input"Kette";c$:fori=1tolen(c$):b$(i)=mid$(c$,i,1):nexti
147 rem ---------------------------------------------------------------------
148 rem berechnung der druck-position
149 rem ---------------------------------------------------------------------
150 y=0:print
160 y=y+1:x=(y-1)*(len(c$)+2):ifx+2+len(c$)> 79then 150
170 print tab(x);:gosub 210:print"  ";
180 gosub 1010
190 ifp4<>0thengoto 160
200 goto 120
207 rem ---------------------------------------------------------------------
208 rem unterprogramm: ausgabe der laufenden zeichenfolge
209 rem ---------------------------------------------------------------------
210 for i=1 to k: print b$(a(i));: next i:return
1007 rem ---------------------------------------------------------------------
1008 rem unterprogramm: aufbau der laufenden zeichenfolge
1009 rem ---------------------------------------------------------------------
1010 p1=n:p2=k:p4=k
1020 fori=1top1:a(k+i)=0:nexti
1030 fori=1top2:a(k+a(i))=1:nexti
1050 p5=a(p4):a(k+p5)=0:p3=p5+1:ifp3>p1then1080
1060 ifa(k+p3)=0then1100
1070 p3=p3+1:ifp3<=p1then1060
1080 p4=p4-1:if1<=p4then1050
1090 p4=0
1100 if p4<>0 then 1130
1110 a(p3)=p3:p3=p3+1:ifp3<=p2then1110
1120 return
1130 a(k+p3)=1:a(p4)=p3:p3=0:p5=0
1140 p4=p4+1:ifp4>p2thenreturn
1150 p3=p3+1:ifa(k+p3)thenp5=p3:goto1150
1160 p5=p5+1:a(p4)=p5:goto1140
1170 end
```

3.

Kommerzielle Programme

Preiskalkulation

Der Handel lebt von der Differenz zwischen Einkaufspreis und Verkaufspreis. Das Programm berechnet bei der Eingabe von zwei der vier Variablen Einkaufspreis, Marge (auch Gewinnspanne genannt), Aufschlag und Verkaufspreis die beiden anderen Variablen. Ferner ist die Eingabe von entweder Marge oder Aufschlag zugelassen, da diese beiden Variablen in einem festen Verhältnis zueinander stehen. Alle anderen Eingabekombinationen werden mit einer Fehlermeldung abgewiesen.

Es gelten folgende Beziehungen zwischen den Variablen:

$$\text{Marge} = \frac{\text{Aufschlag}}{1 + \text{Aufschlag}/100}$$

$$\text{Aufschlag} = \frac{\text{Marge}}{1 - \text{Marge}/100}$$

$$\text{Marge} = \frac{\text{Verkaufspreis} - \text{Einkaufspreis}}{\text{Verkaufspreis}} \star 100$$

$$\text{Aufschlag} = \frac{\text{Verkaufspreis} - \text{Einkaufspreis}}{\text{Einkaufspreis}} \star 100$$

$$\text{Verkaufspreis} = \frac{\text{Einkaufspreis}}{1 - \text{Marge}/100}$$

$$\text{Verkaufspreis} = \text{Einkaufspreis} \star (1 + \text{Aufschlag}/100)$$

$$\text{Einkaufspreis} = \frac{\text{Verkaufspreis}}{1 + \text{Aufschlag}/100}$$

$$\text{Einkaufspreis} = \text{Verkaufspreis} \star (1 - \text{Marge}/100)$$

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Ausgabe der Überschrift
4. Eingabeaufforderungen:

   Einkaufspreis
   Marge
   Aufschlag
   Verkaufspreis

   durch entsprechende Eingabe beantworten
5. Ausgabe der Ergebnisse
6. Beliebige Taste drücken (mit der Taste RUN/STOP wird das Programm beendet). Löschen der Ein- und Ausgabewerte.
7. Rückkehr zur nächsten Eingabe in Schritt 4

Beispiel 1

Eingabe		Ausgabe
Einkaufspreis	? 200	
Marge	? 20	
Aufschlag	?	25
Verkaufspreis	?	250

Beispiel 2

Eingabe		Ausgabe
Einkaufspreis	? 100	
Marge	? 10	
Aufschlag	? 15	
Verkaufspreis	?	

Fehler: Zuviel Variablen

Beispiel 3

Eingabe		Ausgabe
Einkaufspreis	?	
Marge	? 16.6666	
Aufschlag	?	19.999904
Verkaufspreis	?	

Anmerkungen

1. Die Überprüfung und Weiterverarbeitung von Eingabedaten ist nicht nur bei Basic-Programmen von außerordentlicher Wichtigkeit. So müssen Eingaben häufig auf Art (numerisch oder alphanumerisch), Länge oder Vorhandensein eines bestimmten Zeichens (Dezimalpunkt) überprüft werden. Das vorliegende Programm wirft in dieser Hinsicht ein völlig neues Problem auf. Da von den möglichen Eingabewerten maximal zwei eingegeben werden dürfen, ist an sich jede Eingabe für die restlichen Parameter überflüssig und unzulässig. Am besten wäre es also, die Eingabeaufforderung mit der leeren Eingabe (nur die Taste RETURN) zu beantworten. Die leere Eingabe wird von Basic jedoch unterschiedlich und nicht immer im Sinne des Programmierers verstanden.

   Benötigt wird daher eine nichtleere Eingabe, die der leeren Eingabe möglichst ähnlich sieht. Das erreicht man dadurch, daß man in den Begleittext der INPUT-Anweisung eine Cursor-Verschiebung einbaut, die bewirkt, daß nach der (vermeintlich) leeren Eingabe mit RETURN ein Teil des Begleittextes als Eingabe akzeptiert wird. Dieser Textteil kann dann gezielt abgefragt werden. Dieser Plan verliert natürlich seine Wirkung, wenn statt RETURN als erstes die Taste DELETE gedrückt wird, weil damit wieder die Bedingung für eine echte leere Eingabe hergestellt wird.

2. Alle Werte für Marge und Aufschlag sind in Prozent einzugeben.

3. Die Fehlermeldungen:
   Zu wenig Variablen
   Zu viel Variablen
   Unerlaubte Variablen

   erscheinen, wenn keine, mehr als zwei oder unzulässige Kombinationen (z.B. gleichzeitig Marge und Aufschlag) eingegeben werden. Die Verfolgung von möglichen fehlerhaften Eingaben gehört zu jedem ordentlichen Programmkonzept.

Hinweise

1. Zur Steuerung der verteilten Sprunganweisungen (1100-1120) werden im Eingabeteil die beiden Variablen 'z' (Anzahl der Eingaben) und 'y' (Kombination der Eingaben) aufgebaut.

2. Der Ausdruck 't = 6 — 1★(z>2)' (1100) reduziert sich auf das Ergebnis '7' oder '6', je nachdem, ob 'z' größer als 2 ist oder nicht.

```
1001 rem ----------------------------------------------------------------------
1002 rem titel: preiskalkulationen
1003 rem ----------------------------------------------------------------------
1004 rem
1007 rem ----------------------------------------------------------------------
1008 rem text-definitionen
1009 rem ----------------------------------------------------------------------
1010 t$(0)="Einkaufspreis ":t$(1)="Marge          ":t$(2)="Aufschlag      "
1020 t$(3)="Verkaufspreis ":t$(4)="Eingabe":t$(5)="Ausgabe"
1030 t$(6)="Nicht genug ":t$(7)="Zuviel ":t$(8)="Unerlaubte "
1037 rem ----------------------------------------------------------------------
1038 rem initialisation: ueberschrift
1039 rem ----------------------------------------------------------------------
1040 print chr$(147):gosub 3010
1050 r1=8:s1=8:gosub 3000:print t$(4)spc(20)t$(5)
1057 rem ----------------------------------------------------------------------
1058 rem eingabeteil
1059 rem ----------------------------------------------------------------------
1060 z=0:y=0:s1=8:for i=0 to 3:r1=10+i:gosub3000
1070 print t$(i);:input"   ";x$(i):y(i)=val(x$(i))
1080 if x$(i)<>" "thenz=z+1:y=y+i
1090 next i
1100 t=8:on z goto 1110,1120: t=6-1*(z>2):goto 2500
1110 x$(0)=" ":x$(3)=" ":on y goto 2010,2020:goto 2500
1120 on y goto 1170,1190,1150,1180,2000
1147 rem ----------------------------------------------------------------------
1148 rem (marge, aufschlag) = f(einkaufspreis, verkaufspreis)
1149 rem ----------------------------------------------------------------------
1150 if y(0)=0 or y(3)=0 then 2500
1160 y(1)=100*(y(3)-y(0))/y(3):y(2)=100*(y(3)-y(0))/y(0):goto 2100
1167 rem ----------------------------------------------------------------------
1168 rem (verkaufspreis, aufschlag) = f(einkaufspreis, marge)
1169 rem ----------------------------------------------------------------------
1170 y(3)=y(0)/(1-y(1)/100):gosub 2010:goto 2100
1177 rem ----------------------------------------------------------------------
1178 rem (einkaufspreis, aufschlag) = f(verkaufspreis marge)
1179 rem ----------------------------------------------------------------------
1180 y(0)=y(3)*(1-y(1)/100):gosub 2010:goto 2100
1187 rem ----------------------------------------------------------------------
1188 rem (verkaufspreis, marge) = f(einkaufspreis, aufschlag)
1189 rem ----------------------------------------------------------------------
1190 y(3)=y(0)*(1+y(2)/100):gosub 2020:goto 2100
1197 rem ----------------------------------------------------------------------
1198 rem (einkaufspreis, marge) = f(verkaufspreis, aufschlag)
1199 rem ----------------------------------------------------------------------
2000 y(0)=y(3)/(1+y(2)/100):gosub 2020:goto 2100
2006 rem
2007 rem ----------------------------------------------------------------------
```

```
2008 rem aufschlag = f(marge)
2009 rem ----------------------------------------------------------------------
2010 y(2)=y(1)/(1-y(1)/100):if z<>1 then return:goto 2100
2017 rem ----------------------------------------------------------------------
2018 rem marge = f(aufschlag)
2019 rem ----------------------------------------------------------------------
2020 y(1)=y(2)/(1+y(2)/100):if z<>1 then return :goto 2100
2096 rem
2097 rem ----------------------------------------------------------------------
2098 rem ausgabe
2099 rem ----------------------------------------------------------------------
2100 s1=34:for i=0to3:r1=10+i:gosub3000
2110 ifx$(i)<>" "then2130
2120 print y(i)
2130 nexti:goto 2510
2500 r1=18:s1=8:gosub 3000:print "Fehler: "t$(t)"Variablen"
2507 rem ----------------------------------------------------------------------
2508 rem warte- und loesch-schleifen
2509 rem ----------------------------------------------------------------------
2510 get x$:if x$="" then 2510
2520 r1=18:s1=8:gosub 3000:print left$(l$,40)
2530 s1=8:for i=0 to 3:r1=10+i:gosub 3000:print left$(l$,40): nexti
2540 goto 1060
2997 rem ----------------------------------------------------------------------
2998 rem unterprogramm: cursor-steuerung (ueber r1,s1)
2999 rem ----------------------------------------------------------------------
3000 print"▒"left$(cu$,r1)left$(cr$,s1);:return
3007 rem ----------------------------------------------------------------------
3008 rem aufbau der zeichenketten fuer cursor-steuerung und loesch-zeile
3009 rem ----------------------------------------------------------------------
3010 l$=" ":cu$="▒":cr$="▒":fori=1to7:cu$=cu$+cu$:cr$=cr$+cr$:l$=l$+l$:nexti
3020 cu$=left$(cu$,25):cr$=left$(cr$,80):l$=left$(l$,80):return
3030 end
```

Fakturierung

Rechnungen sehen so unterschiedlich aus wie die Firmen, die sie ausschicken. Unterschiede gibt es im Format, in der Menge und Anordnung der Variablen und in der Gewährung von Rabatten und anderen Preisnachlässen. Das vorliegende Programm stellt also nicht den Anspruch, ein Universal-Fakturierungsprogramm zu sein. Vielmehr soll gezeigt werden, wie einfach es ist, ein Grundtableau für die Fakturierung aufzubauen und gleichzeitig für eine benutzerfreundliche Bedienung, einschließlich Fehlerkorrektur, zu sorgen.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Ausgabe der Überschrift
4. Ausgabe der Positionsnummer des aktuellen Postens der Rechnung
5. Eingabeaufforderung:
   Artikel Menge Einzelpreis Rabatt
   durch entsprechende Eingaben beantworten
6. Eingabe auf dem Bildschirm überprüfen. Wenn Eingabe fehlerfrei mit "/" an den Rechner übergeben oder Schritt 8.
7. Ausgabe des Betrags (der sich aus Menge, Einzelpreis und Rabatt errechnet). Rückkehr zu Schritt 5.
8. Aktuelle Eingabe mit der Taste ungleich "/" löschen. Rückkehr zu Schritt 5.
9. Aktuelle Rechnung mit "—" für Artikelbezeichnung beenden.
10. Ausgabe des Endbetrags, der Mehrwertsteuer, des Endbetrags zuzüglich Mehrwertsteuer, des Skonto-Betrags und des Endbetrags abzüglich Skonto.
11. Beliebige Taste drücken (mit der Taste RUN/STOP wird das Programm beendet). Löschen der Ein- und Ausgabewerte.
12. Rückkehr zur nächsten Eingabe in Schritt 5

Beispiel

Nummer	Artikel	Menge	Einzelpreis	Rabatt (%)	Betrag
1	Artikel 1	10	10.50	5	99.75
2	Artikel 2	20	8.50	10	153.00
3	Artikel 3	30	12.35	12	326.04
4	Artikel 4	25	8.75	5	207.81
5	Artikel 5	60	1.50	5	85.50
					872.10
			zzgl. 13% Mehrwertsteuer		113.37
					985.47
Abzüglich 2% Skonto bei Zahlung innerhalb von 8 Tagen					19.71
			Zu zahlender Betrag		965.76

Anmerkungen

1. Auf nichts wird so peinlich genau geachtet wie auf Pfennigbeträge. Alle Einzelbeträge werden also mit der Rundungs-Routine auf Werte mit zwei Nachkommastellen gerundet, bevor sie summiert werden. Das gleiche gilt für die Mehrwertsteuer und Skonto-Beträge.
2. Mehrwertsteuer und Skonto sind im Programm als Konstanten definiert.
3. Für jeden Artikel wird der Rabatt getrennt als Eingabe verlangt. Es kann jedoch auch eine Rabatt-Tabelle verwendet werden, deren Elemente abhängig von der Menge oder des Einzelpreises X Menge (oder einer Kombination von beiden Bedingungen) aufgerufen werden.
4. Es fehlt eine umfassende Überprüfung der Eingabedaten. In dem Programm »Lineare Regression« wird angezeigt, wie diese aussehen kann.

Hinweise

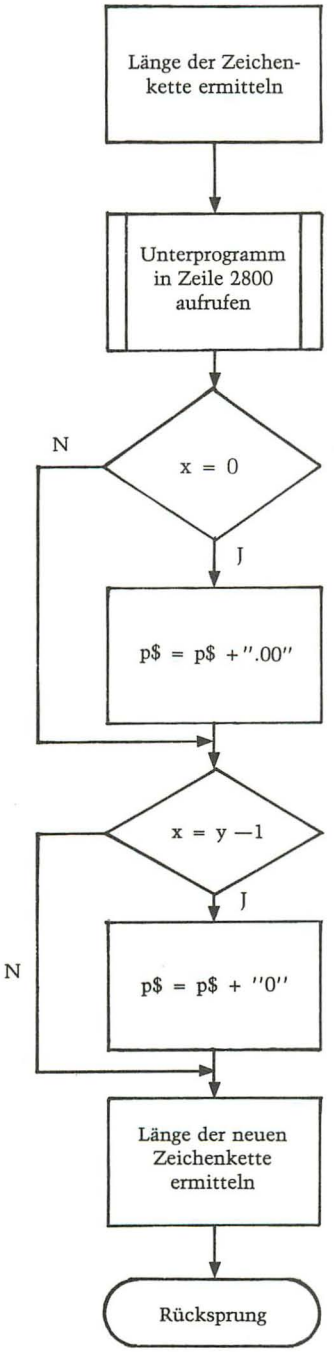
1. Wenn mit der Eingabe '—' für den ersten der fünf Parameter das Ende der aktuellen Rechnung angezeigt wird (1130), wird die Laufvariable auf den Endwert gesetzt, um die Schleife ordnungsgemäß abzuschließen.
2. Die Rundungs-Routine (2600) wird bei der Fakturierung nicht mit dem Parameter für die Anzahl der signifikanten Stellen versorgt, da Geldbeträge immer auf zwei Stellen nach dem Komma gerundet werden.

```
1001 rem ----------------------------------------------------------------------
1002 rem titel: fakturierung
1003 rem ----------------------------------------------------------------------
1004 rem
1007 rem ----------------------------------------------------------------------
1008 rem text-definitionen
1009 rem ----------------------------------------------------------------------
1010 t$(0)="Nummer        ":t$(1)="Artikel       ":t$(2)="Menge         "
1020 t$(3)="Einzelpreis  ":t$(4)="Rabatt (%)    ":t$(5)="        Betrag"
1030 t$(6)="zzgl.":t$(7)="% Mehrwertsteuer":t$(8)="------------"
1040 t$(9)="Abzueglich":t$(10)="% Skonto bei Zahlung innerhalb von 8 Tagen"
1050 u$(0)="Zu zahlender Betrag"
1077 rem ----------------------------------------------------------------------
1078 rem open 1,0 fuer input: initialisierung: ueberschrift
1079 rem ----------------------------------------------------------------------
1080 print chr$(147):gosub 3010:open 1,0:sz(0)=13:sz(1)=2
1090 r1=1:s1=1:gosub 3000:print t$(0)t$(1)t$(2)t$(3)t$(4)t$(5)
1097 rem ----------------------------------------------------------------------
1098 rem eingabeteil
1099 rem ----------------------------------------------------------------------
1100 n=0:z=0
1110 n=n+1
1120 r1=2+n:for i=1 to 5:s1=0:gosub 3000:printn:s1=14+(i-1)*13:gosub3000
1130 input#1,x$(i):y(i)=val(x$(i)):if x$(1)="-"then i=5:x$(5)="/"
1140 next i
1150 if x$(5)<>"/"then s1=0:gosub 3000:print left$(l$,78):goto 1120
1160 if x$(1)="-"then 1190
1170 y=y(2)*y(3)*(1-y(4)/100):gosub 2600:z=z+x:p$=str$(x):gosub 2700
1180 gosub 3000:print" "spc(12-y)p$:goto 1110
1187 rem ----------------------------------------------------------------------
1188 rem summenbildung
1189 rem ----------------------------------------------------------------------
1190 r1=2+n:gosub 2900
1200 r1=3+n:s1=67:gosub 3000:p$=str$(z):gosub 2700:printspc(12-y)p$
1207 rem ----------------------------------------------------------------------
1208 rem mehrwertsteuer
1209 rem ----------------------------------------------------------------------
1210 y=z*sz(0)/100:gosub2600:z1=x:p$=str$(x):gosub 2700
1220 r1=4+n:s1=40:gosub 3000:print t$(6)sz(0)t$(7)spc(14-y)p$
1227 rem ----------------------------------------------------------------------
1228 rem endsumme
1229 rem ----------------------------------------------------------------------
1230 r1=5+n:gosub 2900
```

```
1240 r1=6+n:s1=67:gosub 3000:p$=str$(z+z1):gosub 2700:printspc(12-y)p$
1250 r1=7+n:gosub 2900
1257 rem ---------------------------------------------------------------------
1258 rem skonto
1259 rem ---------------------------------------------------------------------
1260 y=(z+z1)*sz(1)/100:gosub2600:z2=x:z3=z+z1-z2:p$=str$(x):gosub 2700
1270 r1=9+n:s1=10:gosub 3000:print t$(9)sz(1)t$(10)spc(14-y)p$
1280 r1=10+n:gosub 2900
1290 r1=11+n:s1=40:gosub 3000:p$=str$(z3):gosub 2700:printu$(0)spc(20-y)p$
1300 r1=12+n:gosub 2900
2507 rem ---------------------------------------------------------------------
2508 rem warte- und loesch-schleifen
2509 rem ---------------------------------------------------------------------
2510 get x$:if x$="" then 2510
2520 for i=12+n to 3 step -1:r1=i:s1=0:gosub 3000:printleft$(l$,79):nexti
2530 goto 1100
2597 rem ---------------------------------------------------------------------
2598 rem unterprogramm: rundung
2599 rem ---------------------------------------------------------------------
2600 x=int(y*10^2 +0.5)/int(10^2): return
2697 rem ---------------------------------------------------------------------
2698 rem unterprogramm: positionierung des dezimalpunktes
2699 rem ---------------------------------------------------------------------
2700 y=len (p$):gosub 2800
2710 if x=0 then p$=p$+".00"
2720 if x=y-1then p$=p$+"0"
2730 y=len(p$):return
2797 rem ---------------------------------------------------------------------
2798 rem unterprogramm: lokalisierung des dezimalpunktes
2799 rem ---------------------------------------------------------------------
2800 for x=1 to y: if "."=mid$(p$,x,1) then return
2810 next x:x=0: return
2897 rem ---------------------------------------------------------------------
2898 rem unterprogramm: unterstreichung
2899 rem ---------------------------------------------------------------------
2900 s1=0:gosub 3000:printleft$(l$,68)t$(8):return
2997 rem ---------------------------------------------------------------------
2998 rem unterprogramm: cursor-steuerung (ueber r1,s1)
2999 rem ---------------------------------------------------------------------
3000 print"▯"left$(cu$,r1)left$(cr$,s1);:return
3007 rem ---------------------------------------------------------------------
3008 rem aufbau der zeichenketten fuer cursor-steuerung und loesch-zeile
3009 rem ---------------------------------------------------------------------
3010 l$=" ":cu$="▯":cr$="▮":fori=1to7:cu$=cu$+cu$:cr$=cr$+cr$:l$=l$+l$:nexti
3020 cu$=left$(cu$,25):cr$=left$(cr$,80):l$=left$(l$,80):return
3030 end
```

Flußdiagramm

Dezimalteil einer Zahl



Zeile	Anmerkungen
2700	Anzahl Zeichen der eingegebenen Zahl (y)
	Aufruf des Unterprogramms in Zeile 2800 (Suche nach einem Dezimalpunkt)
2710	Zahl ohne Dezimalteil (x = 0)
2720	Zahl mit einer Nachkommastelle (x = y-1)
2730	Neue Länge der Zeichenfolge p$
	Rückkehr aus Unterprogramm

Optimale Bestellmenge

Für die Bestimmung einer optimalen Bestellpolitik in Lagerhaltungsproblemen gibt es eine Reihe von Verfahren. In diesem Beispiel soll die Lösung der klassischen Wilsonschen Formel betrachtet werden. Es wird also vorausgesetzt, daß die Nachfrage konstant ist, Lagerungskosten pro Mengen- und Zeiteinheit gleich sind und keine Fehlmengen zugelassen sind.

Es gilt, die durchschnittlichen Gesamtkosten pro Bestellperiode t

$$C = \frac{h \star Q}{2} + \frac{K \star M}{Q \star T} + \frac{c \star M}{T}$$

zu minimieren.

In der obigen Formel bedeuten

C = Gesamtkosten
K = Beschaffungskosten
h = Lagerungskosten
c = Artikel-Einzelpreis
M = Nachfrage (gesamt)
T = Zeitraum (gesamt)

Für entsprechende Optimierungsverfahren sei auf die Literatur verwiesen. Die Lösung der Formel ergibt

$$Q_{opt} = \sqrt{\frac{2 \star K \star M}{T \star h}}$$ Optimale Bestellmenge

$$t_{opt} = \sqrt{\frac{2 \star K \star T}{h \star M}}$$ Optimale Bestellperiode

Der optimale Bestellpunkt r (Lagerbestand zum Zeitpunkt der Bestellung) errechnet sich aus

$$r_{opt} = \frac{L \star M}{T} - \left[\frac{L}{t} \right] \star Q$$

Hier bedeuten

L = Lieferzeit

$\left[\frac{L}{t} \right]$ = größter ganzzahliger Wert kleiner gleich $\frac{L}{t}$

Die minimalen Gesamtkosten lassen sich schließlich aus der Formel

$$C_{opt} = \sqrt{\frac{2 \star K \star M}{T}} + \frac{c \star M}{T}$$

errechnen.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur
2. Programm starten (run RETURN drücken)
3. Ausgabe der Begleittexte für die Variablen
4. Eingabeaufforderung:

   Beschaffungskosten
   Artikel-Einzelpreis
   Lagerungskosten
   Nachfrage
   Zeitraum
   Lieferzeit

   durch entsprechende Eingaben beantworten
5. Ausgabe der Ergebniswerte

   Bestellmenge
   Bestellperiode
   Bestellpunkt
   Gesamtkosten/Zeit

6. Rückkehr zur nächsten Berechnung in Schritt 3

Anmerkungen

1. Das Programm verwendet eine DATA-Liste, um die Begleittexte für die Variablen zu definieren. In dieser DATA-Liste steht gleichzeitig das (X,Y-) Koordinatenpaar (relativ zur oberen linken Bildschirmecke), das die Position für die Ausgabe des Begleittextes auf dem Bildschirm bezeichnet.
2. Die im Programm "CURL" beschriebene Methode der Cursor-Steuerung wurde verwendet, um die Ein- und Ausgabe auf dem Bildschirm zu steuern.
3. Über einen Schalter (in einer getrennten Eingabe-Operation) bestimmt der Anwender, ob er die Berechnungen fortsetzt oder das Programm beendet.

Beispiel

Beschaffungskosten	40
Artikel-Einzelpreis	.5
Lagerungskosten	.1
Nachfrage (gesamt)	.2400
Zeitraum (gesamt)	12
Lieferzeit	1
Bestellmenge	400
Bestellperiode	2
Bestellpunkt	200
Gesamtkosten/Zeit	140
Weiter/Fertig (w/f)	

Hinweise

1. Die Anweisungen 'dim g(1,11)' (1100), 'data' (1110-1160) und 'read' (1170) stehen in Beziehung zueinander. Die Elemente der DATA-Liste werden mit 'read' gelesen, wonach die für die Cursor-Steuerung verwendeten numerischen Werte im Feld g abgespeichert werden.

```
1001 rem ----------------------------------------------------------------
1002 rem titel: optimale bestellmenge
1003 rem ----------------------------------------------------------------
1004 rem
1097 rem ----------------------------------------------------------------
1098 rem feld-vereinabarungen: open 1,0 fuer input#1: intialisierung
1099 rem ----------------------------------------------------------------
1100 dim g(1,11):gosub1310:printchr$(147):open1,0:restore
1107 rem ----------------------------------------------------------------
1108 rem aufbau der data-liste fuer read-anweisungen
1109 rem ----------------------------------------------------------------
1110 data 10,10,"Beschaffungskosten",11,10,"Artikel-Einzelpreis"
1120 data 12,10,"Lagerungskosten",13,10,"Nachfrage (gesamt)"
1130 data 14,10,"Zeitraum (gesamt)", 15,10,"Lieferzeit"
1140 data 17,10,"Bestellmenge", 18,10,"Bestellperiode"
1150 data 19,10,"Bestellpunkt",20,10,"Gesamtkosten/Zeit"
1160 data 22,10,"Weiter/Fertig (w/f)"
1167 rem ----------------------------------------------------------------
1168 rem read-schleife: ausgabe der data-liste (als hinweis fuer eingabe)
1169 rem ----------------------------------------------------------------
1170 for i=1to11:read r1,s1,a$:gosub1300:printa$:g(0,i)=r1:g(1,i)=s1:nexti
1177 rem ----------------------------------------------------------------
1178 rem eingabe-schleife: eingabe der variablen
1179 rem ----------------------------------------------------------------
1180 for i=1to6:r1=g(0,i):s1=g(1,i)+24:gosub 1300:input#1,x(i):nexti
1187 rem ----------------------------------------------------------------
1188 rem berechnung der ergebnis-werte
1189 rem ----------------------------------------------------------------
1190 y(1)=sqr(2*x(1)*x(4)/(x(3)*x(5))):y(2)=sqr(2*x(1)*x(5)/(x(3)*x(4)))
1200 z=int(x(6)/y(2))
1210 y(4)=sqr(2*x(1)*x(4)*x(3)/x(5))+x(2)*x(4)/x(5):y(3)=x(6)*x(4)/x(5)-y(1)*z
1217 rem ----------------------------------------------------------------
1218 rem ausgabe-teil: ausgabe der ergebnis-werte
1219 rem ----------------------------------------------------------------
1220 for i=1to4:r1=g(0,i+6):s1=g(1,i+6)+24:gosub1300:print y(i):nexti
1227 rem ----------------------------------------------------------------
1228 rem schalter-zeile: nur die eingaben w und f sind erlaubt
1229 rem ----------------------------------------------------------------
1230 r1=g(0,11):s1=g(1,11)+24:gosub1300:printleft$(l$,10):gosub1300:input#1,x$
1240 if x$<>"f"andx$<>"w" then 1230
1250 if x$="f" then  print chr$(147):end
```

```
1257 rem ----------------------------------------------------------------------
1258 rem loesch-routine: loeschen der ein- und ausgabewerte
1259 rem ----------------------------------------------------------------------
1260 fori=1to11:r1=g(0,i):s1=g(1,i)+24:gosub 1300:print left$(l$,20):nexti
1270 goto 1180
1297 rem ----------------------------------------------------------------------
1298 rem unterprogramm: cursor-steuerung (ueber r1,s1)
1299 rem ----------------------------------------------------------------------
1300 print"▤"left$(cu$,r1)left$(cr$,s1);:return
1307 rem ----------------------------------------------------------------------
1308 rem aufbau der zeichenketten fuer cursor-steuerung und loesch-zeile
1309 rem ----------------------------------------------------------------------
1310 l$=" ":cu$="▨":cr$="▮":fori=1to7:cu$=cu$+cu$:cr$=cr$+cr$:l$=l$+l$:nexti
1320 cu$=left$(cu$,25):cr$=left$(cr$,80):l$=left$(l$,80):return
1330 end
```

Zinseszins-Rechnung

Bei unterjährlich-nachschüssiger Verzinsung sind Nominalzins und Effektivzins nicht identisch. Das Jahr ist in m gleiche Zinsperioden unterteilt, und die Zinsgutschrift erfolgt nach jeder dieser Perioden.

Für die beiden gesuchten Größen, Rückzahlungsbetrag und Effektivzins werden die Formeln

$$K = K \star (1 + i/m)^{m \star n}$$

$$p = ((1 + i/m) - 1) \star 100$$

verwendet, wobei

K = Anfangskapital
K = Rückzahlungsbetrag
i = Nominalzins
m = Zinsperioden/Jahr
n = Anzahl Jahre
p = Effektivzins

bedeuten.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Ausgabe der Begleittexte für die Variablen
4. Eingabeaufforderung:

   Anfangs-Kapital
   Nominal-Zinsfuß
   Zinsperioden/Jahre
   Anzahl Jahre

   durch entsprechende Eingaben beantworten
5. Ausgabe der Ergebniswerte

   Rückzahlungs-Betrag
   Effektiv-Zins

6. Rückkehr zur nächsten Berechnung in Schritt 3

Anmerkungen

1. Der äußere Rahmen für dieses Programm ist mit dem des Programms "Lagerhaltung" identisch. Der Unterschied besteht in anderen Begleittexten und natürlich anderen Rechenabläufen. Hier soll gezeigt werden, wie einfach es ist, bewährte Routinen und Programmstrukturen für völlig unterschiedliche Lösungen einzusetzen.

Beispiel

Anfangs-Kapital	6800
Zinsfuß (nominell)	9.5
Zinsperioden/Jahr	4
Anzahl Jahre	10
Rückzahlungs-Betrag	17388.6359
Effektiv-Zins	9.84382816

Weiter/Fertig (w/f)

```
101 rem ----------------------------------------------------------------------
102 rem titel: zinseszins-rechnung
103 rem ----------------------------------------------------------------------
104 rem
107 rem ----------------------------------------------------------------------
108 rem feld-vereinbarungen: open fuer input#1: initialisierung
109 rem ----------------------------------------------------------------------
110 print chr$(147): open 1,0: dim y(1,7):gosub 1310
117 rem ----------------------------------------------------------------------
118 rem aufbau der data-liste fuer read-anweisungen
119 rem ----------------------------------------------------------------------
120 data 10,10,"Anfangs-Kapital"
130 data 10,11,"Zinsfuss (nominell)"
140 data 10,12,"Zinsperioden/Jahr"
150 data 10,13,"Anzahl Jahre"
160 data 10,15,"Rueckzahlungs-Betrag"
170 data 10,16,"Effektiv-Zins"
180 data 10,18,"Weiter/Fertig (w/f)"
187 rem ----------------------------------------------------------------------
188 rem read-schleife: ausgabe der data-liste (als hinweis fuer eingabe)
189 rem ----------------------------------------------------------------------
190 restore
200 for i=1 to 7:read x,y,a$:r1=y:y(0,i)=y:y(1,i)=x:gosub 1300:print a$:nexti
207 rem ----------------------------------------------------------------------
208 rem eingabe-schleife : eingabe der variablen
209 rem ----------------------------------------------------------------------
210 for i=1 to 4:r1=y(0,i):s1=y(1,i)+20:gosub 1300: input#1,x(i):next i
212 rem ----------------------------------------------------------------------
213 rem berechnung und ausgabe der ergebnis-werte
214 rem ----------------------------------------------------------------------
220 p=x(2)/(100*x(3))
230 x(5)=x(1)*(1+p)^(x(3)*x(4)):x(6)=((1+p)^x(3)-1)*100
240 for i=5 to 6: r1=y(0,i):s1=y(1,i)+20:gosub 1300:printx(i):nexti
247 rem ----------------------------------------------------------------------
248 rem schalter-zeile : abfrage auf eingaben 'w' und 'f'
249 rem ----------------------------------------------------------------------
250 r1=y(0,7):s1=y(1,7)+20:gosub1300:printleft$(l$,10):gosub1300:input#1,x$
260 if x$<>"f" and x$<>"w" then 250
267 rem ----------------------------------------------------------------------
268 rem programm-ende :close 1: loeschen des bildschirms
269 rem ----------------------------------------------------------------------
270 if x$="f" then close 1:print chr$(147):end
277 rem ----------------------------------------------------------------------
278 rem loesch-routine
279 rem ----------------------------------------------------------------------
280 for i=1 to 7:r1=y(0,i):s1=y(1,i)+20:gosub 1300:print left$(l$,20):nexti
290 goto 210
1297 rem ---------------------------------------------------------------------
1298 rem cursor-steuerung (ueber r1,s1)
```

```
1299 rem ------------------------------------------------------------------------
1300 print"▒"left$(cu$,r1)left$(cr$,s1);:return
1307 rem ------------------------------------------------------------------------
1308 rem aufbau der zeichenketten fuer cursor-steuerung und loesch-zeile
1309 rem ------------------------------------------------------------------------
1310 l$=" ":cu$="▒":cr$="▌":fori=1to7:cu$=cu$+cu$:cr$=cr$+cr$:l$=l$+l$:nexti
1320 cu$=left$(cu$,25):cr$=left$(cr$,80):l$=left$(l$,80):return
1330 end
```

KURWEG

Dieses Programm ermittelt den kürzesten Weg von einem Ausgangspunkt (Knoten Nr. 1) zu allen anderen Knoten in einem Netzwerk. Das iterative Verfahren berechnet

$f_{k,i}$ = Entfernung von Knoten 1 bis Knoten i nach Schritt k

mit

$f_{1,j} = d_{1,j}$ nach Schritt 1

wobei

$d_{i,j}$ = Entfernung von Knoten i bis Knoten j, wenn die beiden Knoten verbunden sind

$d_{i,j} = \infty$, wenn die beiden Knoten nicht verbunden sind.

Der kürzeste Weg wird dann nach dem Schema

$f_{k,j}$ = minimum $(f_{k-1,i} + d_{i,j})$

berechnet. Nach jeder Iteration wird der augenblicklich kürzeste Weg ausgedruckt. Zuletzt wird abgefragt, ob $d_{i,j} = f_{k,j} - f_{k,i}$, um die Wegstrecken festzustellen, die in der optimalen Lösung enthalten sind.

Bedienungsanweisung

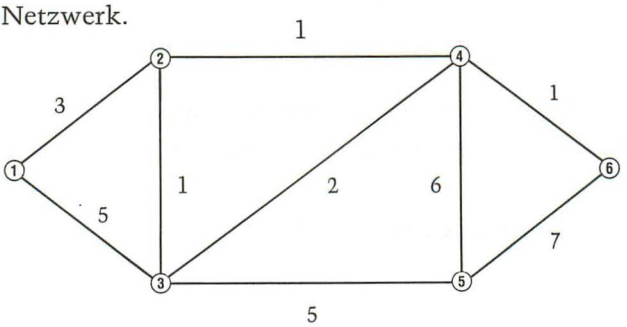
1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Eingabeaufforderung

   Anfangsknoten
   Endknoten
   Länge

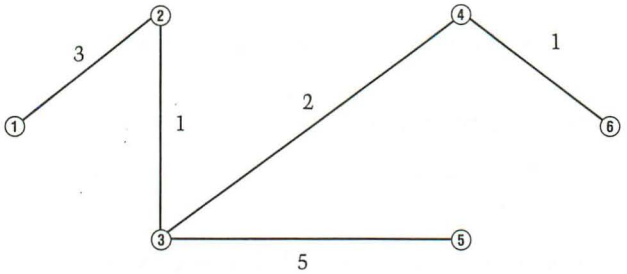
   durch entsprechende Eingaben beantworten.
4. Rückkehr zur nächsten Eingabe in Schritt 3
   oder (bei Eingabe 0)
5. Ausgabe
   der Knoten und Wegstrecken nach der i-ten Iteration
   der Wegstrecken im kürzesten Weg
6. Programmende

Beispiel

Gesucht wird der kürzeste Weg von Knoten 1 zu allen anderen Knoten in dem Netzwerk.



Die optimale Lösung wird durch die Wegstrecke



wiedergegeben.

1. Wegstrecke — Anfangsknoten, Endknoten, Länge ? 1, 2, 3
2. Wegstrecke — Anfangsknoten, Endknoten, Länge ? 1, 3, 5
3. Wegstrecke — Anfangsknoten, Endknoten, Länge ? 2, 3, 1
4. Wegstrecke — Anfangsknoten, Endknoten, Länge ? 2, 4, 4
5. Wegstrecke — Anfangsknoten, Endknoten, Länge ? 3, 4, 2
6. Wegstrecke — Anfangsknoten, Endknoten, Länge ? 3, 5, 5
7. Wegstrecke — Anfangsknoten, Endknoten, Länge ? 4, 5, 6
8. Wegstrecke — Anfangsknoten, Endknoten, Länge ? 4, 6, 1
9. Wegstrecke — Anfangsknoten, Endknoten, Länge ? 5, 6, 7
10. Wegstrecke — Anfangsknoten, Endknoten, Länge ? 0, 0, 0

1-te Iteration

Von Knoten	nach Knoten	Entfernung
1	2	3
1	3	5

2-te Iteration

Von Knoten	nach Knoten	Entfernung
1	2	3
1	3	4
1	4	7
1	5	10

3-te Iteration

Von Knoten	nach Knoten	Entfernung
1	2	3
1	3	4
1	4	6
1	5	9
1	6	8

4-te Iteration

Von Knoten	nach Knoten	Entfernung
1	2	3
1	3	4
1	4	6
1	5	9
1	6	7

5-te Iteration

Von Knoten	nach Knoten	Entfernung
1	2	3
1	3	4
1	4	6
1	5	9
1	6	7

Der kürzeste Weg von Knoten 1 besteht aus:
Wegstrecke 1 — 2
Wegstrecke 2 — 3
Wegstrecke 3 — 4
Wegstrecke 3 — 5
Wegstrecke 4 — 6

Hinweise

Die Felder, die das Netzwerk beschreiben, sind in Zeile 120 dimensioniert. Diese Felder können je nach vorhandenem Speicherplatz anders dimensioniert werden.

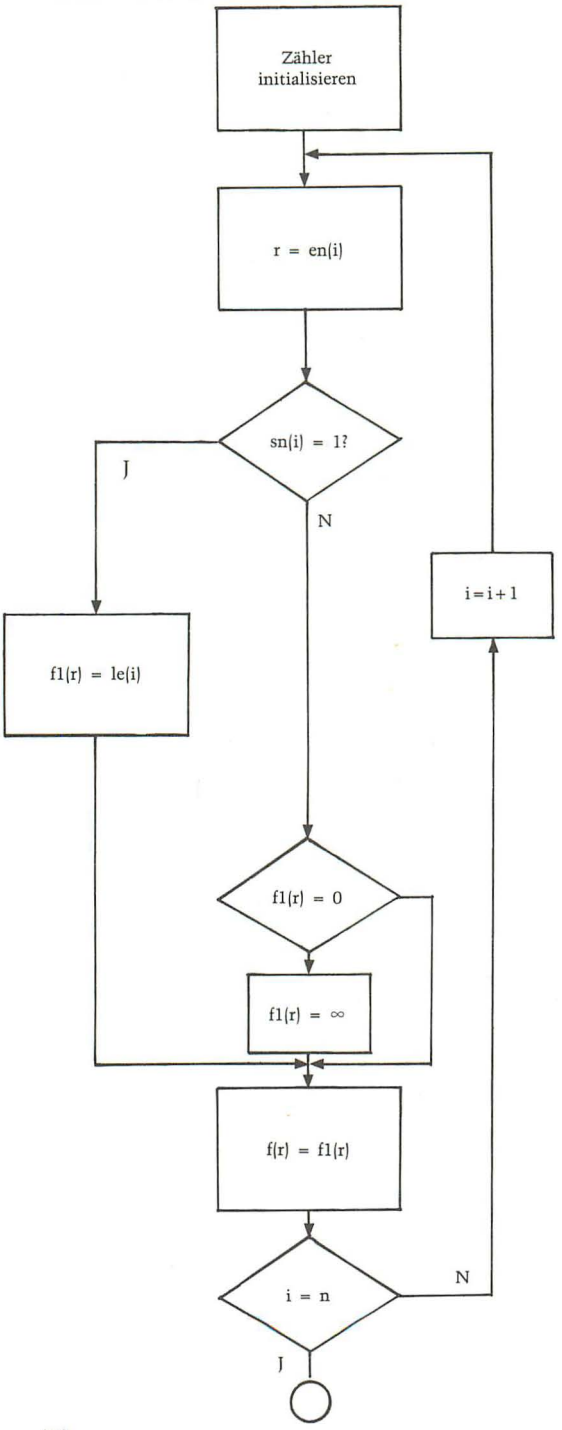
Der Rechner hat keine Möglichkeit, den Wert ∞ explizit wiederzugeben. Es reicht aber, einen im Verhältnis zu den tatsächlichen Wegstrecken sehr großen Wert einzusetzen, um Unendlichkeit zu 'simulieren' (a = 999999999).

```
101 rem ---------------------------------------------------------------------
102 rem titel: kurweg
103 rem ---------------------------------------------------------------------
104 rem
107 rem ---------------------------------------------------------------------
108 rem initialisierung: dimensionierung der felder, konstanten
109 rem ---------------------------------------------------------------------
110 print chr$(147)
120 dim sn(21),en(21),le(21),f(20),f1(20)
130 n0=20:a=999999999:mk=0
137 rem ---------------------------------------------------------------------
138 rem eingabe der wegstrecken-daten
139 rem ---------------------------------------------------------------------
140 for i=1 to n0: a$=str$(i)
150 print spc(3-len(a$))i". Wegstrecke - Anfangsknoten, Endknoten, Laenge ";
160 input sn(i),en(i),le(i):if en(i)>mk then mk=en(i)
170 if sn(i)=0 or en(i)=0 or le(i)= 0 then 190
180 next i
187 rem ---------------------------------------------------------------------
188 rem berechnungsteil: k=1, aufbau der felder f und f1
189 rem ---------------------------------------------------------------------
190 n=i-1:k=1
200 gosub 500
210 for i=1 to n
220 r=en(i): if sn(i)<>1 goto 250
230 print 1spc(10)rspc(10)le(i)
240 f1(r)=le(i): goto 260
250 if f1(r)=0 then f1(r)=a
260 f(r)=f1(r): next i
267 rem ---------------------------------------------------------------------
268 rem berechnungsteil: k>1, aktualisierung der felder f und f1
269 rem ---------------------------------------------------------------------
270 k=k+1
280 for i=1 to n
290 r=sn(i):s=en(i)
300 if f(r)>f1(s)+ le(i) then f(r)=f1(s)+le(i)
310 if f(s)>f1(r)+ le(i) then f(s)=f1(r)+le(i)
320 next i
330 gosub 500
337 rem ---------------------------------------------------------------------
338 rem ausdruck der wegstrecken fuer die laufende iteration
339 rem ---------------------------------------------------------------------
340 for i=2 to mk
350 if f(i)<>a then print 1spc(10)ispc(10)f(i)
360 f1(i)=f(i): next i
370 if k<mk-1 then 270
377 rem ---------------------------------------------------------------------
378 rem ausdruck der kuerzesten wegstrecken von knoten 1
379 rem ---------------------------------------------------------------------
```

```
380 print:print"Der kuerzeste Weg von Knoten 1 besteht aus:":print
390 for i=1 to n:r=sn(i):s=en(i)
400 if le(i)=f(s)-f(r) then print "Wegstrecke "r" - "s
410 next i: end
497 rem ---------------------------------------------------------------------
498 rem unterprogramm: ueberschrift fuer laufende iteration
499 rem ---------------------------------------------------------------------
500 print:print:printk"-te Iteration":print
510 print"Von Knoten   nach Knoten  Entfernung":print
520 return
```

Flußdiagramm

Initialisierung der Felder f und f1



Zeile	Anmerkungen
210	n Wegstrecken
220	r ist der Endpunkt einer Wegstrecke
	Abfrage, ob der Ausgangspunkt einer Wegstrecke gleich 1 ist
240	f1(r) = Weglänge
250	f1(r) = ∞ wenn Knoten r nicht mit Knoten 1 verbunden ist
260	f(r) = f1(r)
	Schleifenende

4.

Statistik

Balkendiagramm

Numerische Daten sollten, sofern sinnvoll und möglich, durch grafische Darstellungsformen ergänzt werden. Eine gute Grafik fördert das Verständnis der Materie und unterbricht die Eintönigkeit von endlosen Zahlenkolonnen.

Nicht alle Grafiken sind gleich leicht zu programmieren. Die Qualität einer Zeichnung hängt von der Auflösung (Anzahl diskrete Punkte pro Flächeneinheit) ab und in welchem Maße schräge Linien wiedergegeben werden können. Ein Stab- oder Balkendiagramm kennt diese Schwierigkeiten jedoch nicht. Auf einer horizontalen Achse werden die in zeitlicher oder anderer Folge angefallenen Summen nebeneinander "aufgetürmt". Diese Form der Präsentation ist anschaulich und gibt einen guten Überblick über die zur Diskussion stehenden Daten.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Eingabeaufforderung:

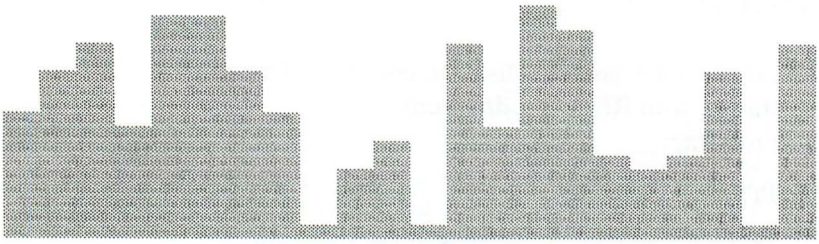
   Anzahl Gruppen

   durch entsprechende Eingaben beantworten
4. Ausgabe des Balkendiagramms
5. Ausgabe der numerischen Werte
6. Rückkehr zur nächsten Eingabe in Schritt 3

Anmerkungen

1. In diesem Programm werden die Variablen (die Länge der Balken) mit dem Zufallszahlen-Generator erzeugt. Viel wichtiger als eine willkürliche Verteilung der Daten ist hier aber die automatische Skalierung, die dabei stattfindet. In einer praktischen Anwendung entstammen die Daten in der Regel einem beliebigen Prozeß. Vor der Ausgabe auf dem Bildschirm muß dann überprüft werden, ob die Anzahl Reihen des Bildschirms für die Ausgabe ausreichen oder ob die Daten nicht erst skaliert werden müssen.
2. Für die Cursor-Positionierung und das Löschen von Bildschirmteilen werden die Programmelemente verwendet, die in dem Programm "CURL" beschrieben sind.

Beispiel



```
 9 12 14  8 16 16 12  9  1  5  7  0 14  8 17 15  6  5  6 12  1 14
Anzahl Gruppen ?
```

Hinweise

1. Die Breite der Balken kann mit der Länge von 'c$' und dem Faktor 'f' verändert werden. Die Höhe des Balkens wird mit der Variablen 'h' gesteuert (120).

```
101 rem ---------------------------------------------------------------------
102 rem titel: balkendiagramm
103 rem ---------------------------------------------------------------------
104 rem
107 rem ---------------------------------------------------------------------
108 rem initialisierung
109 rem ---------------------------------------------------------------------
110 dim a(80): gosub 2110 :m=1
120 printchr$(147):r1=0:s1=0: c$=chr$(230):c$=c$+c$+c$+c$:f=4:h=20
147 rem ---------------------------------------------------------------------
148 rem eingabe: anzahl gruppen
149 rem ---------------------------------------------------------------------
150 r=23:s=9:gosub 2100:print"Anzahl Gruppen"left$(l$,10):s=23: gosub 2100
160 input" ";n
167 rem ---------------------------------------------------------------------
168 rem loesch-routine
169 rem ---------------------------------------------------------------------
170 s=9:for r=0 to 21:gosub 2100:print left$(l$,m*f+1): nextr
197 rem ---------------------------------------------------------------------
198 rem berechnung der balkenlaengen mittels der rnd-funktion
199 rem ---------------------------------------------------------------------
200 fori=1ton:a(i)=int(rnd(0)*h):nexti
207 rem ---------------------------------------------------------------------
208 rem ausgabe des balkendiagramms
209 rem ---------------------------------------------------------------------
210 fori=1 to n: s=i*f+6: gosub 2100
220 forl=1 to a(i): r=20-l: gosub 2100:if a(i)<>0 then print c$
230 next l: nexti
237 rem ---------------------------------------------------------------------
238 rem ausgabe der kolonnen-werte : rueckkehr zum naechsten beispiel
239 rem ---------------------------------------------------------------------
240 r=21: fori=1 to n: s=i*f+9-len(str$(a(i))):gosub 2100:print a(i):nexti
250 m=n:goto 150
2097 rem ---------------------------------------------------------------------
2098 rem unterprogramm: cursor-steuerung (ueber r1,r,s1,s)
2099 rem ---------------------------------------------------------------------
2100 print"▤"left$(cu$,r1+r)left$(cr$,s1+s);:return
2107 rem ---------------------------------------------------------------------
2108 rem aufbau der zeichenketten fuer cursor-steuerung und loesch-zeile
2109 rem ---------------------------------------------------------------------
2110 l$=" ":cu$="▨":cr$="▥":fori=1to7:cu$=cu$+cu$:cr$=cr$+cr$:l$=l$+l$:nexti
2120 cu$=left$(cu$,25):cr$=left$(cr$,80):l$=left$(l$,80):return
2130 end
```

Lineare Regression

Dieses Programm berechnet den Mittelwert, die Varianz, die lineare Regressionskurve und den Korrelationskoeffizienten für n Datensätze (x, y). Zur Ermittlung der statistischen Größen wurden die bekannten Formeln

$$\overline{x} = \frac{\sum x_i}{n}$$

$$\overline{y} = \frac{\sum y_i}{n}$$

$$s_x^2 = \frac{\sum x_i^2 - n\overline{x}^2}{n-1}$$

$$s_y^2 = \frac{\sum y_i^2 - n\overline{y}^2}{n-1}$$

$$a = \frac{\sum x_i y_i - \frac{\sum x_i \sum y_i}{n}}{\sum x_i^2 - \frac{(\sum x_i)^2}{n}}$$

$$b = \overline{y} - a\,\overline{x}$$

$$r = \sqrt{\frac{\left[\sum x_i y_i - \frac{\sum x_i \sum y_i}{n}\right]^2}{\left[\sum x_i^2 - \frac{(\sum x_i)^2}{n}\right]\left[\sum y_i^2 - \frac{(\sum y_i)^2}{n}\right]}}$$

verwendet, wobei

$\overline{x}$ = x-Mittelwert
$\overline{y}$ = y-Mittelwert
s_x^2 = x-Varianz
s_y^2 = y-Varianz
a = Steigung der Regressionsgeraden
b = Schnittpunkt der Geraden mit der Ordinate
r = Korrelationskoeffizient

bedeuten.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Eingabeaufforderung:

   X-Wert Y-Wert

   durch entsprechende Eingaben beantworten oder Schritt 5
4. Rückkehr zur nächsten Eingabe in Schritt 3
5. Nach 10 Wertepaaren oder durch die Eingabe eines Minuszeichens wird die Eingabesequenz unterbrochen. In der Höhe der letzten Eingabe erscheint:
6. Schalterzeile

   f/w/k/e mit

f = fertig	Ausgabe der Statistiken Anzahl Wertepaare X-Mittelwert Y-Mittelwert X-Varianz Y-Varianz Korrelationskoeffizient Regressionskurve Rückkehr zur Schalterzeile
w = weiter	Löschen des aktuellen 10-Blocks und Ausgabe der letzten Eingabe als Referenzpunkt für weitere Eingaben oder Rückkehr zu Schritt 3, wenn der aktuelle 10-Block noch nicht abgeschlossen ist.
k = korrigieren	Eingabe einer Wertepaarnummer im aktuellen 10-Block. Positionsnummern außerhalb des aktuellen Bereichs werden abgewiesen. Löschen des Eintrags in den X-Wert- und Y-Wert-Spalten. Rückkehr zur Schalterzeile. (Durch ein "f" kann die Auswirkung der Wertetilgung sofort untersucht werden.)
e = ende	Programmende

Beispiel

	X-Wert	Y-Wert	
20	10.25	12.45	
21	11.23	14.13	
22	13.56	15.32	
23	15.23	24.23	
24	12.32	16.23	
25	18.13	20.48	
26	16.02	16.78	
27	14.85	19.56	
28	21.06	27.16	
29	14.51	20.26	
30	10.95	9.56	f/w/k/e

Anzahl Wertepaare	30
X-Mittelwert	10.67
Y-Mittelwert	12.64
X-Varianz	27.55
Y-Varianz	46.69
Korrelationskoeff.	.8793
Regressions-Kurve	Y = 1.14 ★ X + 0.47

Anmerkungen

Die Bedeutung des Programms liegt weniger in der richtigen Ermittlung der obigen statistischen Größen als in der Anwendung einiger Programmiertechniken, die zu einem sicheren und effizienten Programmablauf beitragen.

Dazu gehören die Überprüfung numerischer Eingaben, die Aufbereitung der Daten für eine konforme Ausgabe und benutzerfreundliche Aspekte wie schnelle und eindeutige Korrekturmöglichkeiten für falsch eingegebene Werte.

Hinweise

1. Die Routine 'Textvergleich' wird verwendet (1690), um festzustellen, ob der Eingabewert einen Dezimalpunkt enthält.

```
1001 rem ---------------------------------------------------------------------
1002 rem titel: lineare regression
1003 rem ---------------------------------------------------------------------
1004 rem
1007 rem ---------------------------------------------------------------------
1008 rem feld-vereinbarungen: open 1,0 fuer input#: initialisierung
1009 rem ---------------------------------------------------------------------
1010 dimy(1,11): r1=2: s1=6: gosub1720:printchr$(147):open1,0
1020 a$="X-Wert          Y-Wert":r=-2: s=9: gosub 1710: printa$
1030 m=0: n=x(1)
1037 rem ---------------------------------------------------------------------
1038 rem eingabe-routine
1039 rem ---------------------------------------------------------------------
1040 m=m+1: if m>10 then 1160
1050 r=m :s=0:gosub1710:printn+m;left$(l$,30)
1060 s=10:gosub1710:input#1,x$:ifx$="-"goto 1160
1070 x=0:gosub1650:ifx=1then1050
1080 y(0,m)=val(x$)
1090 s=20:gosub1710: print left$(l$,10):gosub 1710
1100 input#1,x$
1110 x=0:gosub1650:ifx=1then1090
1120 y(1,m)=val(x$)
1130 r=m:s=4:gosub1710:print left$(l$,30):j=m:gosub 1550
1140 i=m: p=1:gosub 1620 :goto 1040
1160 m=m-1
1167 rem ---------------------------------------------------------------------
1168 rem schalter-zeile : nur die eingaben f/w/k/e sind zugelassen
1169 rem ---------------------------------------------------------------------
1170 r=m:s=36:gosub1710:print"f/w/k/e"+left$(l$,10);:s=45:gosub1710:input#1,z$
1180 ifz$= "e" then 1335
1190 ifz$= "f" then 1340
1200 ifz$= "k" then 1260
1210 ifz$<>"w" then 1170
1217 rem ---------------------------------------------------------------------
1218 rem w = weiter
1219 rem ---------------------------------------------------------------------
1220 s=36:gosub 1710: print left$(l$,10): if m<10 goto 1040
1230 s=0:for r=11 to 0 step-1 :gosub1710:printleft$(l$,50):nextr
1240 r=0: s=0:gosub 1710:printn+m:j=m: gosub 1550
1250 goto 1030
1257 rem ---------------------------------------------------------------------
1258 rem k = korrigieren: eingabe einer werte-positionsnummer
1259 rem ---------------------------------------------------------------------
1260 r=m+1:s=36:gosub1710:print"Nr: ";:input#1,z$ :z=int(val(z$))
1270 if z<=n or z>n+m then 1170
1280 s=0:for r=m+2 to z-n step-1: gosub1710: printleft$(l$,50):nextr
1290 p=-1: for i=z-n to m: gosub 1620: y(0,i)=y(0,i+1):y(1,i)=y(1,i+1): nexti
1300 p=1: for j=z-n to m-1:if m-z+n=0 then 1320
1310 i=j:gosub1620 :r=j:s=0:gosub1710:printj+n:gosub1550
```

```
1320 next j
1330 m=m-1: goto 1170
1332 rem ----------------------------------------------------------------------
1333 rem e = ende: schliessen der tastatur als datei: loeschen der anzeige
1334 rem ----------------------------------------------------------------------
1335 close 1:print chr$(147):end
1337 rem ----------------------------------------------------------------------
1338 rem f = fertig: berechnung und ausgabe der statistischen groessen
1339 rem ----------------------------------------------------------------------
1340 n1=x(1)
1350 a=(x(6)-(x(2)*x(3)/n1))/(x(4)-x(2)*x(2)/n1):y=a:z=2:gosub1580:a=x
1360 xm=x(2)/n1: ym=x(3)/n1
1370 b=ym-a*xm:y=b:z=2:gosub1580:b=x
1380 vx=(x(4)-x(2)*x(2)/n1)/(n1-1):vy=(x(5)-x(3)*x(3)/n1)/(n1-1)
1390 kk=(x(6)-(x(2)*x(3)/n1))
1400 kk=kk/sqr((x(4)-x(2)*x(2)/n1)*(x(5)-x(3)*x(3)/n1))
1410 s=1:r=12:gosub1710:print"Anzahl Wertepaare  ";n1
1420 s=1:r=13:gosub1710:print"X-Mittelwert       "
1430 y=xm:z=2:gosub1580:s=24:gosub 1570
1440 s=1:r=14:gosub1710:print"Y-Mittelwert       "
1450 y=ym:z=2:gosub1580:s=24:gosub 1570
1460 s=1:r=16:gosub1710:print"X-Varianz          "
1470 y=vx:z=2:gosub1580:s=24:gosub 1570
1480 s=1:r=17:gosub1710:print"Y-Varianz          "
1490 y=vy:z=2:gosub1580:s=24:gosub 1570
1500 s=1:r=19:gosub1710:print"Korrelationskoeff."
1510 y=kk:z=4:gosub1580:s=24:gosub 1570
1520 s=1: r=20:gosub1710:print"Regressions-Kurve   ";left$(l$,20)
1530 s=21:r=20:gosub1710:print"Y = "a" * X +"b
1540 goto 1170
1547 rem ----------------------------------------------------------------------
1548 rem unterprogramm: ausgabe-routine fuer x/y-werte
1549 rem ----------------------------------------------------------------------
1550 s=11:i=0:gosub1560:s=26:i=1:gosub1560: return
1560 x$=str$(y(i,j)):gosub1690:s=s-k:gosub1710: printx$: return
1567 rem ----------------------------------------------------------------------
1568 rem unterprogramm: ausgabe-routine fuer statistik-werte
1569 rem ----------------------------------------------------------------------
1570 x$=str$(x):gosub1690:s=s-k:gosub1710: printx$: return
1577 rem ----------------------------------------------------------------------
1578 rem unterprogramm: rundungs-routine fuer statistik-werte
1579 rem ----------------------------------------------------------------------
1580 x=int(y*10↑z+.5)/10↑z:return
1617 rem ----------------------------------------------------------------------
1618 rem summenbildung : x, y, x*x, y*y, x*y  (p = 1 oder -1)
1619 rem ----------------------------------------------------------------------
1620 x(1)=x(1)+p*1:x(2)=x(2)+p*y(0,i):x(3)=x(3)+p*y(1,i)
1630 x(4)=x(4)+p*(y(0,i)*y(0,i)):x(5)=x(5)+p*(y(1,i)*y(1,i))
1640 x(6)=x(6)+p*(y(0,i)*y(1,i)): return
```

```
1647 rem ---------------------------------------------------------------------
1648 rem unterprogramm: abfrage nach gueltiger zahlen-eingabe
1649 rem ---------------------------------------------------------------------
1650 y$=x$: if left$(x$,1)<>"-" then y$=" "+x$
1660 if str$(val(x$))<> y$ thenx=1
1670 return
1687 rem ---------------------------------------------------------------------
1688 rem unterprogramm: auffindung des dezimalpunktes
1689 rem ---------------------------------------------------------------------
1690 for k=1 to len(x$):if"."=mid$(x$,k,1)thenreturn
1700 nextk:k=len(x$)+1:x$=x$+".":return
1707 rem ---------------------------------------------------------------------
1708 rem unterprogramm: cursor-steuerung (ueber r1,s1,r,s)
1709 rem ---------------------------------------------------------------------
1710 print"▤"left$(cu$,r1+r)left$(cr$,s1+s);:return
1717 rem ---------------------------------------------------------------------
1718 rem aufbau der zeichenketten fuer cursor-steuerung und loesch-zeile
1719 rem ---------------------------------------------------------------------
1720 l$=" ":cu$="▨":cr$="▥":fori=1to7:cu$=cu$+cu$:cr$=cr$+cr$:l$=l$+l$:nexti
1730 cu$=left$(cu$,25):cr$=left$(cr$,80):l$=left$(l$,80):return
1740 end
```

5.

Mathematik

Primzahlen

Primzahlen gehören zu den natürlichen Zahlen (1, 2, 3 ... N) und haben die Eigenschaft, daß sie nur durch 1 oder sich selber geteilt werden können. Primzahlen sind gewissermaßen die Bausteine aller natürlichen Zahlen, da sich jede Zahl als Produkt der Primzahlpotenzen zusammensetzt, die sich in die Zahl dividieren lassen. So entspricht die Zahl 60 beispielsweise dem Produkt $2^2 \cdot 3 \cdot 5$.

Primzahlen haben die Menschheit schon seit Jahrtausenden beschäftigt und es war Euklid, der bereits im dritten vorchristlichen Jahrhundert auf ebenso einfache wie elegante Art beweisen konnte, daß die Menge der Primzahlen unbegrenzt ist. Bei Primzahlen interessieren erzeugende Funktionen (wie etwa die Eulersche Formel $N = x^2 + x + 41$, die unterschiedliche Primzahlen für $x = 1, 2, \ldots 39$ erzeugt), Primzahlpaare (11, 13) und vieles mehr.

Das Programm verwendet die Methode des Eratosthenes, um, ausgehend von den beiden bekannten Primzahlen 2 und 3, eine Tabelle von Primzahlen aufzubauen. Der Algorithmus macht von der Tatsache Gebrauch, daß eine Zahl N, die keine Primzahl ist, einen Divisor d hat, für den $1 < d \leq \sqrt{N}$ gilt.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Eingabeaufforderung
   Anzahl der Primzahlen
4. Ausgabe
   Längste primzahlenlose Sequenz
   Primzahlen
5. Programmende

Beispiel

Anzahl Primzahlen (a > 1 und a < 6107)? 100

Längste primzahlenlose Sequenz 523 - 541 : 17 Elemente

2	3	5	7	11	13	17	19
23	29	31	37	41	43	47	53
59	61	67	71	73	79	83	89
97	101	103	107	109	113	127	131
137	139	149	151	157	163	167	173
179	181	191	193	197	199	211	223
227	229	233	239	241	251	257	263
269	271	277	281	283	293	307	311
313	317	331	337	347	349	353	359
367	373	379	383	389	397	401	409
419	421	431	433	439	443	449	457
461	463	467	479	487	491	499	503
509	521	523	541				

Hinweise

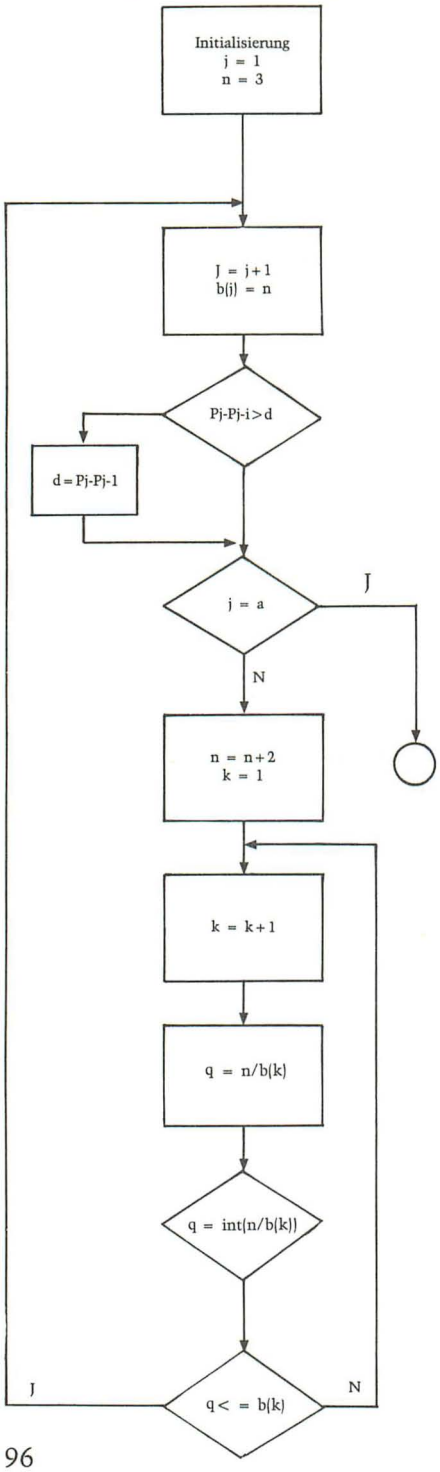
Mit dem Ausdruck 'int(fre(0)/5)—10' (120) wird der Speicherplatz berechnet, der für die Tabelle für die Primzahlen zur Verfügung steht. Die Funktion 'fre(0)' schreibt den freien Speicherplatz (in Byte) zurück und dieser Wert wird durch 5 dividiert, da eine Gleitkommazahl 5 Byte belegt. Von diesem Ergebnis wird dann noch 10 subtrahiert, um etwas Programmspeicher in Reserve zu haben. Der frei verfügbare Speicher kann auch durch 2 geteilt werden, wenn Primzahlen als Ganzzahlen im Bereich 2 — 32767 erwünscht sind.

Die Primzahlen werden mit den Anweisungen in 260 bis 290 in tabellarischer Form ausgedruckt. Diese Anweisungsfolge kann für alle Tabellen mit 'a' Elementen verwendet werden. Die Anzahl der Spalten wird mit 'b', der Abstand zwischen den Kolonnen mit dem ersten Teil des Arguments der SPC-Anweisung (280) festgelegt.

```
101 rem ---------------------------------------------------------------------
102 rem titel: primzahlen
103 rem ---------------------------------------------------------------------
104 rem
107 rem ---------------------------------------------------------------------
108 rem loeschen des bildschirms
109 rem ---------------------------------------------------------------------
110 print chr$(147)
117 rem ---------------------------------------------------------------------
118 rem eingabe
119 rem ---------------------------------------------------------------------
120 print "Anzahl Primzahlen (a > 1 und  a <"int(fre(0)/5)-10")";
130 input a
137 rem ---------------------------------------------------------------------
138 rem feldvereinbarung: aufbau der grundstellung
139 rem ---------------------------------------------------------------------
140 dim b(a):d=0
150 b(1)=2:j=1:n=3
160 j=j+1:b(j)=n: if n-b(j-1)>d then d=n-b(j-1):c=j
170 if j=a then 240
180 n=n+2:k=1
190 k=k+1: q=n/b(k): if q=int(n/b(k)) then 180
200 if q<=b(k) then 160
210 goto 190
237 rem ---------------------------------------------------------------------
238 rem ausgabe
239 rem ---------------------------------------------------------------------
240 print:print "Laengste primzahlenlose Sequenz ";
250 print b(c-1)" - "b(c)" : "b(c)-b(c-1)-1"  Elemente":print:print
260 b=7:for i=1 to a step b+1:if i> a-b then b=a-i
270 for j=i to i+b
280 print spc(6-len(str$(b(j))));b(j);:next j
290 print :next i
300 end
```

Flußdiagramm

Berechnung der Primzahlen



Zeile	Anmerkungen
150	Erste Primzahl ist 2 Primzahlenzähler j Zweite Primzahl ist 3
160	Nächste Primzahl
	Ermittlung der längsten primzahlenlosen Sequenz (wird für die Berechnung der Primzahlen nicht benötigt)
170	Gewünschte Anzahl von Primzahlen erreicht?
180	Zu prüfende Zahl (n) um 2 inkrementieren; k gleich 1 setzen
190	k inkrementieren q = n/b(k)
	Abfrage, ob Rest gleich Null
200	Abfrage, ob Rest < = b(k)

Nullstellenbestimmung nach Newton-Raphson

In vielen Fällen lassen sich Gleichungen der Form

$$f(x) = 0$$

auf analytische Weise lösen. Wenn analytische Methoden nicht ausreichen, werden numerische Verfahren zur Bestimmung der Nullstellen einer Funktion verwendet.

Die Methode von Newton-Raphson besteht darin, die Gleichung $f(x) = 0$ in die Form $g(x) = x$ mit

$$g(x) = x - \frac{f(x)}{f'(x)}$$

zu bringen. Mit dem Näherungswert p_0 wird dann eine Sequenz $\{P_n\}$ nach dem Iterationsschema $p_n = g(p_{n-1})$ erzeugt. Diese Sequenz konvergiert zu dem Wert $p = x$ und einer Lösung der Gleichung $f(x) = 0$, wenn $f(p) = 0$ und $f(x)$ stetig in $x = p$ ist.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. f(x) und f'(x) definieren (siehe Anmerkung 1)
3. Programm starten (run RETURN drücken)
4. Eingabeaufforderungen

   Delta
   Iterationen
   Anfangswert

   durch entsprechende Eingaben beantworten
5. Ausgabe
   Lösung gefunden (nach n Iterationen)
   Funktion hat eine Nullstelle bei x = xxxxx
   bzw.
   Keine Lösung gefunden (nach N Iterationen)
6. Rückkehr zur nächsten Eingabe in Schritt 4

Anmerkungen

1. Um mittels der Newton-Raphson-Methode eine Nullstelle einer Funktion ausfindig zu machen, muß sowohl die Funktion selber als auch ihre erste Ableitung bekannt sein. Die beiden Funktionen werden als f(x) und g(x) definiert. Dazu wird die Anweisung 'def fn' (hier in Zeilen 110 und 120) verwendet.
2. Mit 'Delta' wird die Genauigkeit der gesuchten Lösung festgelegt. Die Lösung (sofern eine existiert) wird bei größerem Delta u.U. schneller gefunden, hat dann aber vielleicht nicht die volle Genauigkeit.
3. Mit 'Iterationen' wird ein oberer Grenzwert für die Anzahl der Iterationen angegeben. Damit wird vermieden, daß das Programm sich 'verabschiedet', wenn die Funktion nicht den Bedingungen für eine Lösung genügt.
4. Wenn der Algorithmus nach der Eingabe von 'Anfangswert' eine Lösung findet, wird diese angezeigt und zur erneuten Eingabe zurückverzweigt, um gegebenenfalls mit anderen Anfangswerten weitere Nullstellen zu untersuchen.

Beispiel 1

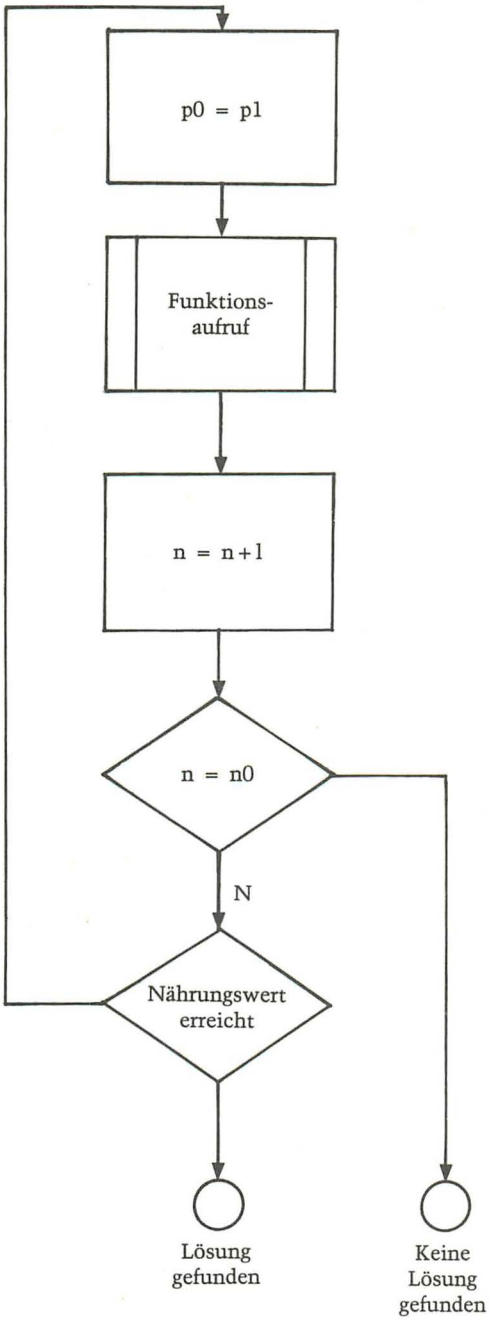
```
110 def fn f(x) = x*x — 4*x +3
120 def fn g(x) = 2*x — 4

Delta          ? .000001
Iterationen    ? 100
Anfangswert    ? —10

Lösung gefunden (nach 8 Iterationen)

Funktion f(x) hat eine Nullstelle bei x = .999999999

Delta          ? .000000000001
Iterationen    ? 100
Anfangswert    ? 1000000000000

Lösung gefunden (nach 46 Iterationen)

Funktion f(x) hat eine Nullstelle bei x = 3
```

Beispiel 2

```
110 def fn f(x) = tan(x)
120 def fn g(x) = 1/(cos(x) * cos(x))

Delta          ? .000001
Iterationen    ? 10
Anfangswert    ? 12

Lösung gefunden (nach 4 Iterationen)

Funktion f(x) hat eine Nullstelle bei x = 12.5663706

Delta          ? .000000000001
Iterationen    ? 10
Anfangswert    ? 2.5

Lösung gefunden (nach 5 Iterationen)

Funktion f(x) hat eine Nullstelle bei x = 3.14159266
```

```
101 rem ----------------------------------------------------------------
102 rem titel: nullstellenbestimmung nach newton-raphson
103 rem ----------------------------------------------------------------
104 rem
107 rem ----------------------------------------------------------------
108 rem definition der funktion und ihrer ersten ableitung
109 rem ----------------------------------------------------------------
110 def fn f(x) = x*x - 4*x +3
120 def fn g(x) = 2*x - 4
130 def fn h(x) = x - fn f(x)/fn g(x)
137 rem ----------------------------------------------------------------
138 rem initialisierung des bildschirms: aufruf der programmbeschreibung
139 rem ----------------------------------------------------------------
140 print chr$(147): gosub 250
147 rem ----------------------------------------------------------------
148 rem eingabe von genauigkeit, iterationen und 1. naeherungswertes
149 rem ----------------------------------------------------------------
150 input"Delta        ";d1
160 input"Iterationen ";n0
170 input"Anfangswert ";p1: n=0
177 rem ----------------------------------------------------------------
178 rem berechnung des naechsten naeherungswertes: inkrementierung von n
179 rem ----------------------------------------------------------------
180 p0=p1:p1= fn h(p0):n=n+1:if n=n0 then 230
190 if d1 < abs(p0-p1) then 180
197 rem ----------------------------------------------------------------
198 rem ausgabe wenn d1 < abs(p0-p1)
199 rem ----------------------------------------------------------------
200 print:print "Loesung gefunden (nach"n"Iterationen)"
210 print:print "Funktion f(x) hat eine Nullstelle bei x = "p0:print:print
220 goto 150
227 rem ----------------------------------------------------------------
228 rem ausgabe wenn n = n0
229 rem ----------------------------------------------------------------
230 print:print "Keine Loesung gefunden (nach"n0"Iterationen)"
240 goto 150
247 rem ----------------------------------------------------------------
248 rem programm-kurzbeschreibung
249 rem ----------------------------------------------------------------
250 print "Um mittels der Newton-Raphson-Methode eine Nullstelle einer"
260 print "Funktion ausfindig zu machen, muss sowohl die Funktion sel-"
270 print "ber als auch ihre erste Ableitung bekannt sein. Die beiden"
280 print "Funktionen werden als f(x) und g(x) definiert. Dazu wird die"
290 print "Anweisung 'def fn' (hier in Zeilen 110 und 120) verwendet."
300 print:return
310 end
```

Flußdiagramm

Berechnung des Näherungswertes	Zeile	Anmerkungen
	180	p0 = letzter Näherungswert bzw. der vom Anwender eingegebene Anfangswert
		p1 = fn h(p0)
		Iterationszähler inkrementieren
		Maximale Anzahl von Iterationen durchgeführt?
	190	Näherungswert mit gewünschter Genauigkeit?

6.

Spiel- und Lehrprogramme

Wortrate-Spiel

Das im Englischen unter dem Namen "Hangman" bekannte Spiel ist für wenigstens zwei Spieler konzipiert. Ein Spieler gibt, ohne dabei von den anderen beobachtet zu werden, ein Wort in den Rechner ein. Die anderen Spieler haben dann die Gelegenheit, das versteckte, in seiner Länge durch Bindestriche angezeigte Wort zu erraten. Mit jedem falschen oder bereits einmal erfolgreich eingesetzten Buchstaben wird der Bau eines zehnteiligen Galgens fortgesetzt. Nach zehn falschen Buchstaben werden die fehlenden Buchstaben in dem Wort eingesetzt. Der Rechner geht dann an den Spielanfang zurück. Bei einem Treffer werden alle Bindestriche, die für den Buchstaben stehen, durch diesen ersetzt. Das Spiel gewinnt an Reiz, wenn die Eingaben auf bestimmte Themen, historische Persönlichkeiten oder Fremdwörter beschränkt werden.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Eingabe des Rätselwortes. Der Rechner akzeptiert nur Buchstaben des lateinischen Alphabets, um Wörter bis zu einer Länge von 10 Zeichen aufzubauen. Bei kürzeren Wörtern muß die Eingabe durch ein RETURN abgeschlossen werden. Für jeden eingegebenen Buchstaben erscheint auf dem Bildschirm ein Bindestrich. Die Anzahl der Bindestriche zeigt somit die Länge des Rätselwortes an.
4. Suche nach dem Rätselwort

   a. Eingabe eines im Wort enthaltenen Buchstabens
   Der gefundene Buchstabe ersetzt alle Bindestriche, die für ihn im Rätselwort stehen. Der Buchstabe wird danach gesperrt, d. h. er wird fortan als nicht im Wort enthalten betrachtet.

   b. Eingabe eines nicht im Wort enthaltenen Buchstabens
   Der Buchstabe wird als ungültig ausgegeben und der Galgen um ein Stück weiter gebaut.
5. Bei erfolgreicher Suche erscheint die Meldung "Gerettet", andernfalls die Meldungen "Gehangen", wobei die noch fehlenden Buchstaben im Rätselwort eingefügt werden.
6. Rückkehr zur Eingabe des nächsten Rätselwortes in Schritt 3

Anmerkungen

1. Für die Ausgabe des Piktogramms (Galgen mit Pendant) werden zwei Unterprogramme verwendet, deren Wirkung durch eine DATA-Liste gesteuert wird. Die Liste besteht aus:

   a. Unterprogrammnummer (für die ON ... GOSUB-Anweisung)
   b. X-Koordinate
   c. Y-Koordinate
   d. Wiederholungsfaktor für das auszugebende Zeichen
   e. Zeichen, das n-mal ab (X,Y) auf den Bildschirm ausgegeben werden soll

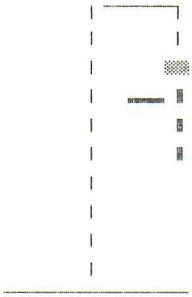
   Die Unterprogrammnummer bestimmt, ob die Ausgabe in der vertikalen oder horizontalen Richtung erfolgt.

2. Für die Cursor-Steuerung wird die POKE-Methode verwendet. Dazu wird Zelle 216 mit der Reihennummer und Zelle 198 mit der Spaltennummer geladen.

Hinweise

1. Bei der Eingabe des Rätselwortes werden alle nicht alphabetischen Eingaben abgewiesen (270).

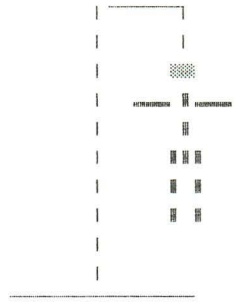
H E N K E R



p y t h a g o r a s G e r e t t e t

Ungueltige Buchstaben: e i u n m f k

H E N K E R



p h l e g m a G e h a n g e n

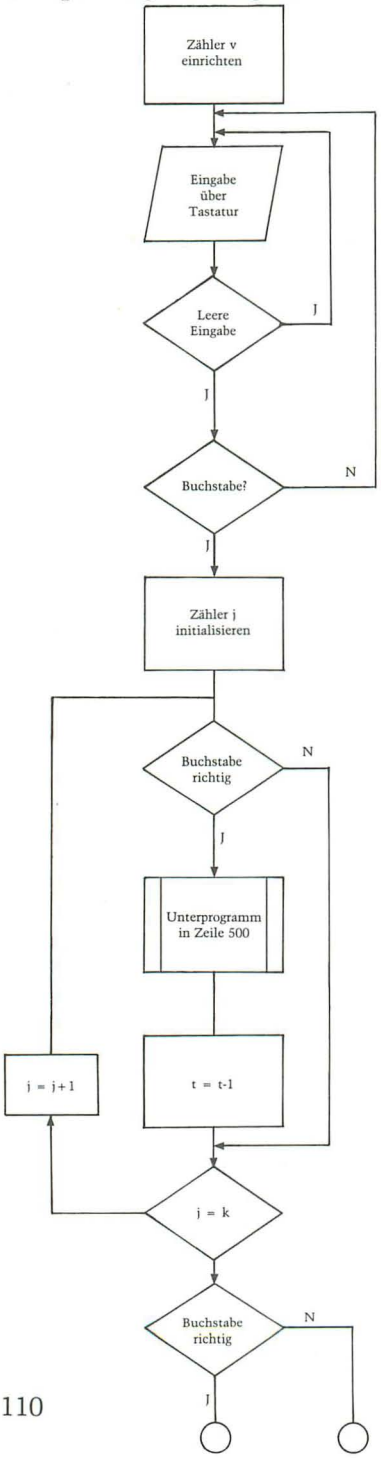
Ungueltige Buchstaben: i o u n b d r o k s

```
101 rem ----------------------------------------------------------------
102 rem wortrate-spiel
103 rem ----------------------------------------------------------------
104 rem
107 rem ----------------------------------------------------------------
108 rem aufbau der data-liste
109 rem ----------------------------------------------------------------
110 data 1,40,2,10,229
120 data 2,40,3,6,227
130 data 2,32,13,15,227
140 data 1,47,2,2,229
150 data 2,45,5,2,230
160 data 1,47,5,3,225
170 data 2,42,6,3,249
180 data 2,47,6,3,249
190 data 1,46,7,3,225
200 data 1,48,7,3,225
210 printchr$(147)
220 restore:s=8:r=3:gosub1000:print"H E N K E R"
227 rem ----------------------------------------------------------------
228 rem eingabe-schleife verstecktes wort
229 rem ----------------------------------------------------------------
230 r=14:k=10:fori=1to10
240 gety$:ify$=""then 240
250 ifi=1thenprintchr$(147)
260 ifasc(y$)=13thenk=i-1:i=10:goto 290
270 ifasc(y$)>90orasc(y$)<65then 240
280 s=i*2+6:gosub1000:y1$(i)=y$:print"-"
290 nexti
297 rem ----------------------------------------------------------------
298 rem eingabe-schleife fuer buchstaben
299 rem ----------------------------------------------------------------
300 u=0:t=k
310 v=t:gety$:ify$=""then 310
320 ifasc(y$)>90orasc(y$)<65then 310
330 forj=1tok:ify$=y1$(j)then gosub 500:t=t-1:y1$(j)="*"
340 nextj:if v<>t then 390
347 rem ----------------------------------------------------------------
348 rem read-anweisung bei ungueltigem buchstaben
349 rem ----------------------------------------------------------------
350 read a,x,y,n,z:u=u+1
360 on a gosub 800,900
370 r=16:s=8:gosub 1000:print"Ungueltige Buchstaben:"
380 s=u*2+31:gosub 1000:printy$
387 rem ----------------------------------------------------------------
388 rem bei t=0 sind alle buchstaben gefunden
389 rem ----------------------------------------------------------------
390 if t=0 then s=33:r=14:gosub 1000:print"G e r e t t e t":goto 220
400 if u<10 then 310
```

```
407 rem ------------------------------------------------------------------
408 rem bei u=10 ist das spiel verloren
409 rem ------------------------------------------------------------------
410 for j=1 to k:gosub 500: nextj
420 s=33:r=14:gosub 1000:print"G e h a n g e n":goto 220
497 rem ------------------------------------------------------------------
498 rem unterprogramm: ausgabe von buchstaben
499 rem ------------------------------------------------------------------
500 r=14:s=j*2+6:gosub1000:ify1$(j)<>"*" then printy1$(j)
510 return
797 rem ------------------------------------------------------------------
798 rem unterprogramme fuer die ausgabe des piktogramms
799 rem ------------------------------------------------------------------
800 r=y:s=x:gosub1000:for l=1ton:r=y+l:gosub1000:printchr$(z):nextl:return
900 r=y:s=x:gosub1000:for l=1ton:s=x+l:gosub1000:printchr$(z):nextl:return
997 rem ------------------------------------------------------------------
998 rem unterprogramm fuer die cursor-steuerung
999 rem ------------------------------------------------------------------
1000 poke216,r:print"▨":poke198,s:return
1010 end
```

Flußdiagramm

Überprüfung der Eingabe



Zeile	Anmerkungen
310	v = Anzahl der gesuchten Buchstaben Zeichen von der Tastatur einlesen (y$)
	Leere Eingabe?
320	Alphabetisches Zeichen?
330	Wort enthält k Buchstaben
	Buchstabe in Rätselwort?
	Unterprogramm in Zeile 500 aufrufen (Ausgabe des Buchstaben im Rätselwort)
	t dekrementieren Gefundenen Buchstaben durch '★' ersetzen
340	
	Buchstabe in Rätselwort?

Bagels

Das Programm "Bagels" ist eine der ersten Versionen des heute unter der Bezeichnung "Master-Mind" bekannten und zurecht beliebten Code-Knacker-Spiels. Mit einer festen Länge von drei Ziffern ohne Wiederholungen stellt Bagels die vielleicht optimale Form des Spiels dar. Die Ergebnisse einer Spielserie werden summiert und nach jedem Spiel als kurze Statistik ausgegeben. Nach neun erfolglosen Versuchen bricht der Rechner das Spiel ab, das dann auch mit neun bewertet wird. Der Durchschnitt für einen guten Spieler sollte um fünf Versuche pro Spiel liegen.

Der Rechner bedient sich folgender Kommentare, um den Spieler in der Suche nach der versteckten Zahl zu unterstützen:

bagels	— alle Ziffern falsch
pico	— eine Ziffer richtig, aber an falscher Stelle
pico-pico	— zwei Ziffern richtig, beide an falscher Stelle
pico-pico-pico	— drei Ziffern richtig, alle an falscher Stelle
fermi	— eine Ziffer richtig und an richtiger Stelle
fermi-fermi	— zwei Ziffern richtig, beide an richtiger Stelle
pico-fermi	— zwei Ziffern richtig, eine an richtiger und eine an falscher Stelle
pico-pico-fermi	— drei Ziffern richtig, eine an richtiger und zwei an falscher Stelle

Bei Ziffernwiederholungen erscheint die Meldung "Keine Wiederholungen". Eine ungültige Eingabe dieser Art wird nicht als Versuch gewertet.

Erfolg wird unterschiedlich belohnt. Der Spieler erhält den Text:

Pfiffikus	— bei vier oder weniger Versuchen
Nicht schlecht	— bei fünf Versuchen
Also doch	— bei sechs Versuchen
Na endlich	— bei sieben Versuchen
Kaum zu glauben	— bei acht Versuchen
Eins vor Zwölf	— bei neun Versuchen

Beim neunten (und erfolglosen) Versuch wird der Rechner ungehalten und bricht das Spiel mit dem Kommentar "Wieder nicht — jetzt reicht's aber" ab.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Ausgabe der Programm-Kurzbeschreibung
4. Eingabe einer dreistelligen Zahl
5. Kommentar des Rechners zu der eingegebenen Zahl
   Wiederholung der Schritte 4 und 5, bis die gesuchte Zahl gefunden ist oder neun Versuche erfolgt sind
6. Ausgabe der Statistiken: Anzahl Spiele
   Anzahl Versuche/Spiel
7. Rückkehr zur nächsten Eingabe in Schritt 4 oder Programmabbruch mit STOP

Anmerkungen

1. Im Programm wird die POKE-Methode für die Cursor-Steuerung eingesetzt. Dazu wird Zelle 216 mit der Reihennummer und Zelle 198 mit der Spaltennummer geladen.
2. Die "versteckte" Zahl wird mit der RND-Funktion des Rechners erzeugt und in drei Ganzzahlen zerlegt. Die Zufälligkeit der Zahl (wichtig in diesem Spiel) wird dadurch verbessert (gegenüber der Methode, drei einstellige Ganzzahlen mit der RND-Funktion zu erzeugen).
3. Zur Übernahme der Zahl des Spielers wird die GET-Funktion verwendet. Da GET zeichenweise arbeitet, ist auch die Versuchszahl schon in einer Form, die die erforderlichen Abfragen und Überprüfungen unterstützt.
4. Das Programm benutzt für den Kommentar des Rechners einen errechneten Textzugriff. Ein Zähler wird anfangs auf Null gesetzt und dann für jedes "Pico" um 1 und jedes "Fermi" um 4 erhöht. Damit ergibt sich für jede mögliche Antwort des Rechners eine andere Zahl im Bereich 0 bis 8. Da es keine Pico-Fermi-Kombination gibt, die sich auf 7 summiert, wird diese Textposition für "Keine Wiederholungen" verwendet.

Beispiele

258 Fermi
269 Fermi-Fermi
149 Fermi
307 Pico
239 Fermi-Fermi
279 Also doch

Anzahl Spiele 1
Anzahl Versuche/Spiel 6

852 Bagels
963 Fermi
941 Pico
467 Pico-Fermi
064 Nicht schlecht

Anzahl Spiele 2
Anzahl Versuche/Spiel 5.5

Hinweise

1. Die RND-Funktion wird verwendet, um eine dreistellige Zahl zu generieren (190).

```
101 rem ---------------------------------------------------------------------
102 rem titel: bagels
103 rem ---------------------------------------------------------------------
104 rem
107 rem ---------------------------------------------------------------------
108 rem feld-vereinbarungen: bildschirm-ausgaben: initialisierung
109 rem ---------------------------------------------------------------------
110 dim x(3),y(3),z(3),t$(16):print chr$(147): gosub 1000:n1=0:n2=0
117 rem ---------------------------------------------------------------------
118 rem text-definitionen
119 rem ---------------------------------------------------------------------
120 t$(16)="                                "
130 t$(0)="Bagels":t$(1)="Pico":t$(2)="Pico-Pico":t$(3)="Pico-Pico-Pico"
140 t$(4)="Fermi":t$(5)="Pico-Fermi":t$(6)="Pico-Pico-Fermi"
150 t$(7)="Keine Wiederholungen":t$(8)="Fermi-Fermi":t$(9)="Pfiffikus"
160 t$(10)="Nicht schlecht":t$(11)="Also doch":t$(12)="Na endlich"
170 t$(13)="Kaum zu glauben":t$(14)="Eins vor Zwoelf"
180 t$(15)="Wieder nicht - jetzt reicht's aber"
187 rem ---------------------------------------------------------------------
188 rem programm-hauptteil
189 rem ---------------------------------------------------------------------
190 n=0:x(0)=int(rnd(0)*1000):x(1)=int(x(0)/100):x(2)=int((x(0)-x(1)*100)/10)
200 x(3)=x(0)-x(1)*100-x(2)*10
210 fori=1to3:y(i)=x(i):nexti:gosub400:ifp<>0then190
220 n=n+1
230 fori=1 to 3
240 gety$:ify$="" then 240
250 ifz<>7 then 270
260 gosub 450:printt$(16)
270 if n=1andi=1 then print chr$(147);
280 if asc(y$)>57orasc(y$)<48 then 240
290 gosub450:printy$;:y(i)=val(y$):nexti
300 gosub 400:z=7:ifp<>0 then 340
310 z=0:j1=1:j2=2:j3=3:gosub 420
320 j1=2:j2=3:j3=1:gosub 420
330 j1=3:j2=1:j3=2:gosub 420:ifz>8orn>8 goto 360
340 print" "+t$(z);:ifz<>7 goto 220
350 goto 230
360 m=n-(z<>12):if n<5 then m=4
370 print" "+t$(m+5)
380 n1=n1+1:n2=n2+n
390 n=n+2:i=5:gosub 450:print"Anzahl Spiele "n1
395 n=n+1:i=5:gosub 450:print"Anzahl Versuche/Spiel "n2/n1:goto 190
397 rem ---------------------------------------------------------------------
398 rem unterprogramm zur erkennung von ziffer-wiederholungen
399 rem ---------------------------------------------------------------------
400 p=0:ify(1)=y(2)ory(1)=y(3)ory(2)=y(3)thenp=1
410 return
417 rem ---------------------------------------------------------------------
```

```
418 rem unterprogramm zur berechnung des ergebniswertes der eingabezahl
419 rem ---------------------------------------------------------------------
420 ifx(j1)=y(j2)orx(j1)=y(j3)thenz=z+1
430 ifx(j1)=y(j1)thenz=z+4
440 return
447 rem ---------------------------------------------------------------------
448 rem unterprogramm zur positionierung des cursors
449 rem ---------------------------------------------------------------------
450 poke216,8+n:print"▓": poke198,30+i:return
500 end
997 rem ---------------------------------------------------------------------
998 rem programm-beschreibung - ausgabe auf dem bildschirm
999 rem ---------------------------------------------------------------------
1000 print"Eine vom Rechner erzeugte, dreistellige Zahl ist zu erraten."
1010 print"Ziffernwiederholungen - etwa 233 - sind nicht erlaubt. Der Rechner"
1020 print"gibt nach jeder Eingabe Auskunft, inwiefern die eingegebene Zahl"
1030 print"mit der vom Rechner erzeugten Zahl uebereinstimmt. Dabei gilt:"
1040 print:print"          bagels - alle Ziffern falsch"
1050 print"          pico   - eine Ziffer richtig, aber an falscher Stelle"
1060 print"          fermi  - eine Ziffer richtig und an richtiger Stelle"
1070 print:print"Beispiel:":print:print
1075 print"Die 'versteckte' Zahl  216  des Rechners"
1080 print"     und Ihre Eingabe  261":print
1090 print"ergeben die Meldung 'pico-pico-fermi' weil 6 und 1 zwar in der"
1100 print"Zahl enthalten sind, aber an falscher Stelle stehen, waehrend die"
1110 print"Ziffer 2 auch noch an der richtigen Stelle steht. Ein Spiel wird"
1111 print"nach neun erfolglosen Versuchen abgebrochen. Das Programm wird"
1120 print"mit der Taste RUN/STOP beendet."
1130 return:end
```

Matrix-Lehrgang

Eine der leistungsfähigsten und damit wichtigsten Basic-Anweisungen ist die DIM-Anweisung, die dazu dient, Felder (auch Arrays oder Matrizen genannt) zu definieren. Das vorliegende Programm veranschaulicht das Konzept einer Matrix.

Der Anwender gibt in Antwort auf die Eingabeaufforderungen des Rechners die Dimension der Matrix als r Reihen mal s Spalten ein. Danach wird die Matrix auf dem Bildschirm ausgegeben, wobei in Anlehnung an die DIM-Anweisung bei der Numerierung der Reihen und Spalten bei Null begonnen wird.

Zu Beginn sind alle Elemente der Matrix auf Null gesetzt. Durch Eingabe einer Reihen- und Spaltennummer kann jeder Wert in der Matrix verändert werden, wobei die neue Belegung sofort auf dem Bildschirm angezeigt wird.

Bedienungsanweisung

1. Programm laden (von Kassette, Diskette oder über Tastatur)
2. Programm starten (run RETURN drücken)
3. Eingabeaufforderungen:

   Anzahl Reihen:
   Anzahl Spalten:

   durch entsprechende Eingaben beantworten
4. Ausdruck der (leeren) Matrix
5. Eingabeaufforderungen:

   r, s, A (r, s)

   durch entsprechende Eingabe beantworten
6. Ausgabe des mit (r, s) angesteuerten Elements der Matrix
7. Rückkehr zur nächsten Eingabe in Schritt 5

Anmerkungen

1. Damit bei der Ausgabe der Matrix keine Bildverschiebungen passieren, ist die Anzahl der Spalten auf 12 und die der Reihen auf 14 zu beschränken. Für die Werte der einzelnen Elemente seien Ganzzahlen im Bereich 0 bis 9999 zu verwenden.
2. Das Programm verwendet einfache Schleifenmethoden, um die Position eines Elements auf dem Bildschirm anzusteuern. Bezugspunkt ist die obere linke Bildschirmecke, die mit chr$(147) erreicht wird, bzw. die erste Reihe der Matrix, die immer die gleiche Entfernung zur oberen Bildschirmkante einhält.

Beispiel

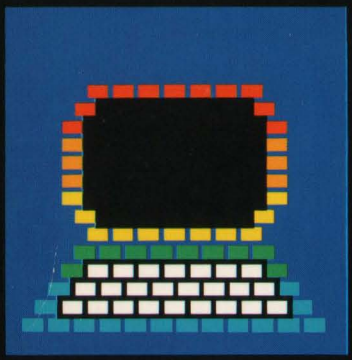
Anzahl Reihen: ? 10
Anzahl Spalten: ? 12

	S 0	S 1	S 2	S 3	S 4	S 5	S 6	S 7	S 8	S 9	S 10	S 11	S 12
R 0	11	0	0	0	0	0	0	0	0	0	0	0	0
R 1	0	22	0	0	0	0	0	0	0	0	0	0	0
R 2	0	0	33	0	0	0	0	0	0	0	0	0	0
R 3	0	0	0	44	0	0	0	0	0	0	0	0	0
R 4	0	0	0	0	55	0	0	0	0	0	0	0	0
R 5	0	0	0	0	0	66	0	0	0	0	0	0	0
R 6	0	0	0	0	0	0	77	0	0	0	0	0	0
R 7	0	0	0	0	0	0	0	88	0	0	0	0	0
R 8	0	0	0	0	0	0	0	0	99	0	0	0	0
R 9	0	0	0	0	0	0	0	0	0	100	0	0	0
R 10	0	0	0	0	0	0	0	0	0	0	111	0	9999

r, s, A (r, s)? 10, 12, 9999

```
101 rem ----------------------------------------------------------------
102 rem titel: matrix-lehrgang
103 rem ----------------------------------------------------------------
104 rem
107 rem ----------------------------------------------------------------
108 rem initialisierung: eingabe der anzahl reihen und spalten
109 rem ----------------------------------------------------------------
110 printchr$(147):dima(20,20)
120 input"Anzahl Reihen:  ";r
130 input"Anzahl Spalten: ";s
140 ifr>14ors>12 then 120
150 print:print
160 forj=0 to s
167 rem ----------------------------------------------------------------
168 rem ausgabe der spalten-beschriftung
169 rem ----------------------------------------------------------------
170 printtab(j*5+6);"S";:nextj:print
180 forj=0 to s
190 printtab(j*5+6);mid$(str$(j),2);:nextj:print:print
197 rem ----------------------------------------------------------------
198 rem ausgabe der reihen-beschriftung: ausgabe der feldelemente
199 rem ----------------------------------------------------------------
200 fori=0 to r
210 forj=0 to s:if j=0 thenprint"R ";mid$(str$(i),2);
220 printtab(j*5+5);a(r,s);
230 nextj:print
240 nexti:print
247 rem ----------------------------------------------------------------
248 rem eingabe der element-position (r,s) und des wertes (a(r,s))
249 rem ----------------------------------------------------------------
250 input"r,s,A(r,s)";x,y,z:ifx=<randy=<s then 270
257 rem ----------------------------------------------------------------
258 rem fehlermeldung: r oder s > anzahl reihen oder spalten
259 rem ----------------------------------------------------------------
260 print"Eingabefehler":print"[illegible]":goto 250
267 rem ----------------------------------------------------------------
268 rem berechnung der bildschirm-position bei legaler r,s-eingabe
269 rem ----------------------------------------------------------------
270 print"                    "chr$(19)
275 fori=0to6+x:print"[illegible]";:nexti
280 printtab(y*5+5);"     [illegible]"z:a(x,y)=z
287 rem ----------------------------------------------------------------
288 rem berechnung der bildschirm-position fuer erneute eingabe
289 rem ----------------------------------------------------------------
290 fori=1 to r-x+1:print"[illegible]";:nexti:goto 250
300 end
```

BASIC ohne Probleme
Band 2: Übungen



Dieses Übungsbuch ist die Ergänzung zu dem im gleichen Verlag erschienenen Lehrbuch »BASIC ohne Probleme/Band 1: Unterweisung«. Es enthält 20 Routinen und Programme, die vor allem den Anfänger in der Programmierung unterstützen sollen, so rasch wie möglich seinen Rechner erfolgreich zum Einsatz zu bringen.

Die Programme setzen sich aus einer Beschreibung, Bedienungsanweisungen, Beispielen, Anmerkungen und reichlich mit Kommentaren versehenen Programmprotokollen zusammen. Die Information wird, sofern sinnvoll und erforderlich, durch Tips, Erläuterungen und Flußdiagramme ergänzt. Die Programme haben in erster Linie die Aufgabe, allgemeine Programmiertechniken zu veranschaulichen. Daher wurde auf besondere, wenn auch gewinnbringende Tricks und Techniken in dieser Sammlung verzichtet.
Die Betonung liegt vielmehr auf den alltäglichen und daher besonders wichtigen Problemen in der Programmierung: Die Behandlung und Verarbeitung von Eingabedaten, die Steuerung des Cursors, die Verwendung von Unterprogrammen und die Programmierung von Fehlerausgängen.

Die Programme in diesem Buch sind in fünf Rubriken unterteilt: drei allgemeine Routinen, die auch in anderen Programmen zur Anwendung kommen, fünf allgemeine Programme, fünf kommerziell-technische Anwendungen (Preiskalkulation, Fakturierung, optimale Bestellmenge, Zinseszins-Rechnung) zwei Statistikprogramme, zwei Mathematikprogramme und drei (hoffentlich spannende) Lehr- und Spielprogramme. Alle Programme können direkt in einen cbm-Rechner Modell 8032 eingegeben und gestartet werden. Für andere Rechner sind fallweise Anpassungen vorzunehmen.

Markt&Technik Verlagsgesellschaft mbH
ISBN 3-922120-26-1